UNDERSTANDING AND MITIGATING PRIVACY RISK IN MACHINE LEARNING SYSTEMS

BY

YUNHUI LONG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Doctoral Committee:

Professor Carl A. Gunter, Chair
Professor ChengXiang Zhai
Professor Bo Li
Professor Reza Shokri

ABSTRACT

Recent years have witnessed a rapid development in machine learning systems and a widespread increase of machine learning applications. However, with the widespread adoption of machine learning, privacy issues have emerged. This thesis studies the privacy risk in modern machine learning systems in two ways.

First, we improve the understanding on machine learning privacy through attacks and measurements. Due to the increasing complexity and lack of transparency of state-of-art machine learning models, it is challenging to understand what information a model learns from its training data and whether the information could be leaked through the model's predictions. Therefore, we design various attacks to infer different information from machine learning models trained on sensitive data. By analyzing the performance of these attacks, we get a better understanding on the privacy risk of sharing these models.

Second, we propose different levels of protection mechanisms to balance between privacy and data utility. We divide the use of sensitive data in a modern machine learning system into three levels based on the trade-off between data utility and privacy protection.

At the first level, we consider data with high utility requirement and relatively low privacy protection, such as system logs with heterogeneous data of high dimensionality. This type of data is very sensitive to noise injection, making it challenging to achieve strong privacy guarantee without incurring great loss on data utility. To address this problem, we propose empirical protections based on hypothesis tests. Our approach uses various hypothesis tests to identify potential information leakage from the data and adds the minimum amount of noise sufficient to mitigate the identified risks. Although this approach does not provide strong theoretical guarantee, it allows users to share their data with higher confidence and with minimum utility loss.

At the second level, we consider sensitive data that need to be shared for general purposes. For example, datasets containing personal photos can be used in a wide range of applications including face recognition, human pose extraction, and mood detection. However, these photos are also extremely sensitive since they contain a lot of privacy information. For this type of data, it is important to maintain a proper balance between privacy and data utility. On the one hand, due to the sensitive nature of the data, it is necessary to apply rigorous privacy protections such as differential privacy. On the other hand, to allow multiple applications to use the released data, the privacy protection mechanisms need to preserve the original data distribution to the maximum extent possible. Based on these requirements, we design a

novel approach G-PATE for training a scalable differentially private data generator, which can be used to produce synthetic datasets with strong privacy guarantee while preserving high data utility.

At the third level, we consider sensitive data that are useful for specific applications. For this type of data, it is often not necessary to share the original dataset. Instead, data owners can share differentially private machine learning models tailored to the need of the applications. By only sharing the models, we limit the use of the sensitive data to the approved applications while improving model utility under the same privacy guarantee. As an example, in this thesis, we propose the first differentially private graph convolutional network (DP-GCN). By guaranteeing edge-differential privacy, DP-GCN allows users to analyze graph-structured data without leaking the sensitive connection information, such as private real-life connections in social networks.

To my parents, for their love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my Ph.D. advisor, Professor Carl Gunter. Carl has taught me a lot about doing research and being an independent researcher. I am especially grateful for his continuous support and encouragement. He has always supported me to explore my ideas and to pursue the directions I am passionate in. His advises and encouragement helped me build confidence in myself and guided me to overcome many obstacles throughout my Ph.D. study. Thanks to Carl for his continuous guidance and support.

Second, I am fortunate to have Prof. Carl Gunter, Prof. ChengXiang Zhai, Prof. Bo Li, and Prof. Reza Shokri serve on my thesis committee. I would like to thank them for their insightful questions, helpful suggestions, and valuable feedback. Their suggestions on earlier versions of this work have inspired me to think more broadly about the problem and the impact of this work.

In addition, I would like to thank my brilliant collaborators, including Carl Gunter, Xiaofeng Wang, Bo Li, Kai Chen, Yang Zhang, Vincent Bindschaedler, Le Xu, and Lei Wang. This thesis cannot be done without the kind assistance they all provided to me. In particular, thanks to Xiaofeng for providing invaluable guidance on the projects and on doing research in general, thanks to Bo for all the inspiring discussions and kind support.

Next, I want to express my appreciation to my Illinois Security Lab labmates: Muhammad Naveed, Aston Zhang, Vincent Bindschaedler, Soteris Demetriou, Wei Yang, Güliz Seray Tuncay, Qi Wang, Avesta Hojjati, Hyun Bin Lee, Allyson Kaminsky, and Xiaojun Xu. I would like thank them for the stimulating discussions and the pleasant working environment. Besides, I would also like to thank all my friends for their kind support and company. In particular, thanks to Silu Huang, Le Xu, and Mengjia Yan for the great time we had together.

Finally, I would like to thank my father Xiaorong Long, my mother Lanling Li, and my boyfriend Chenxing Wang for their unconditional love and support. Their love gives me the courage to pursue whatever I want, knowing that they will always be there by my side when I need them.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1
1.1 The Foundation: Understanding Privacy Risk under Pragmatic Adversarial Models 2
1.2 Three Levels of Privacy Protections 2
1.3 Thesis Contributions and Organizations 4

CHAPTER 2 RELATED WORK 6
2.1 Privacy Attacks on Machine Learning Models 6
2.2 Differential Privacy and Rényi Differential Privacy. 7
2.3 Empirical Privacy Protection Mechanisms 9
2.4 Trade-Offs between Privacy and Utility 9

CHAPTER 3 A PRAGMATIC APPROACH TO MEMBERSHIP INFERENCES ON MACHINE LEARNING MODELS 11
3.1 Adversary Model 12
3.2 Pragmatic Membership Inference Attack 15
3.3 Evaluation 24
3.4 Discussion 33
3.5 Conclusions 34

CHAPTER 4 TOWARDS MEASURING MEMBERSHIP PRIVACY 36
4.1 Problem Statement 38
4.2 Differential Training Privacy 40
4.3 Case Studies 42
4.4 Protections against Indirect Membership Attacks 52
4.5 Reducing DTP 59
4.6 Discussion 60
4.7 Open Questions 63
4.8 Conclusions 64

CHAPTER 5 BLACK-BOX PROPERTY INFERENCE ATTACKS ON MACHINE LEARNING MODELS 65
5.1 Problem Statement 69
5.2 Methodology 71
5.3 Evaluation 76
5.4 Conclusion 82

CHAPTER 6 A HYPOTHESIS TESTING APPROACH TO SHARING LOGS WITH CONFIDENCE 83
6.1 Log Indistinguishability 85
6.2 Indistinguishability Tests 89
6.3 Protections with Log Obfuscation 95
6.4 Case Studies 97
6.5 Conclusion 106

CHAPTER 7 SCALABLE DP GENERATIVE MODEL VIA PATE 107
7.1 The G-PATE Method 109
7.2 Theoretical Guarantees 113
7.3 Experimental Evaluation 115
7.4 Conclusion 120

CHAPTER 8 DIFFERENTIALLY PRIVATE GRAPH CONVOLUTIONAL NEURAL NETWORKS 121
8.1 Differentially Private GCN 122
8.2 Experiments 128
8.3 Conclusion 133

CHAPTER 9 CONCLUSION 134

REFERENCES 136

CHAPTER 1: INTRODUCTION

Machine learning has been applied to a wide range of applications such as face recognition [1], autonomous driving [2], and medical diagnoses [3, 4]. However, as machine learning systems begin to play a more and more important role in our life, there is an increasing concern on the potential privacy risks associated with these systems. From k-anonymity [5] to differential privacy [6], different privacy definitions and mechanisms have been proposed to prevent privacy leakage from sensitive datasets or machine learning models. Yet, privacy protections often come at a cost of reducing data utility, and there have been a long-lasting discussion around balancing this trade-off.

This thesis studies the privacy-utility trade-off from a multi-level perspective. The contribution of the thesis can be summarized in the privacy pyramid shown in Figure 1.1. As the foundation of the pyramid, we study the privacy risk of machine learning systems through a series of attacks under pragmatic adversarial models. These studies aim to provide a better understanding of privacy risk in real-world scenarios. Based on this understanding, we divide privacy protections into three levels with a distinct trade-off between privacy and utility requirements at each level. At the first level, we consider data with high utility requirements and relatively low privacy protection. At the second level, we design methods to share data for general uses under strong privacy protections. At the third level, we focus on privately sharing data for specific uses.

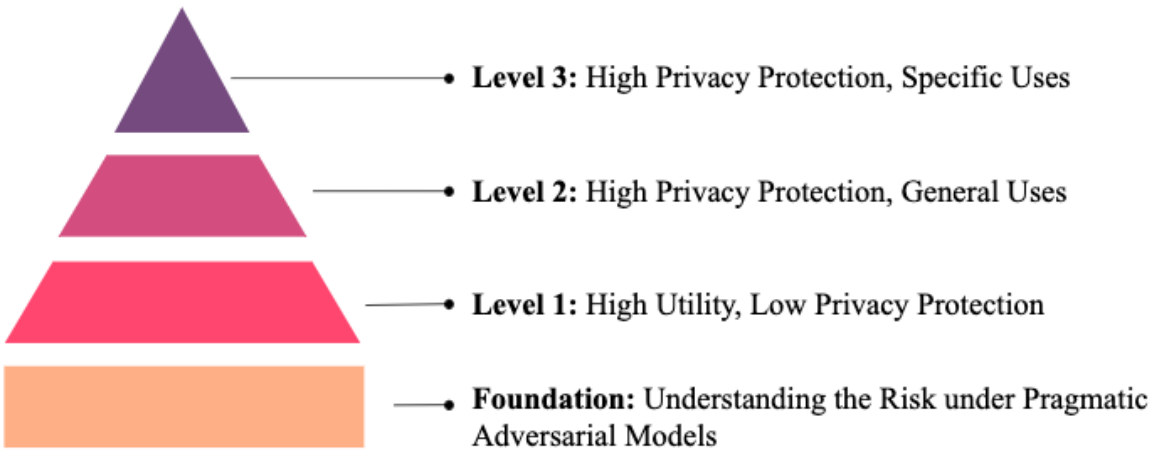


Figure 1.1: The Privacy Pyramid.

1.1 THE FOUNDATION: UNDERSTANDING PRIVACY RISK UNDER PRAGMATIC ADVERSARIAL MODELS

Understanding the privacy risk associated with sharing a dataset or a machine learning model is a foundational step towards balancing privacy and utility. Specifically, we aim to understand privacy risks from two aspects: *(1) What information can be inferred from the released dataset or model? (2) What extra information or resources is needed to make a correct inference?*

To answer the first question, we take two approaches to analyzing the information leakage: the attack-based approach and the test-based approach. In an attack-based approach, we model the inference problem as a game between the adversary and the data owner, and evaluate privacy risk as the adversary's advantage or accuracy in inferring the sensitive information. This approach has been commonly used in different attacks against machine learning models [7, 8]. Although the attack-based approach provides a clear understanding on the risk of a specific attack, it often fails to give a whole picture of the general privacy risk. Therefore, we design the test-based approach to more efficiently understand the risk of various attacks. Instead of performing the actual attacks, we design light-weighted hypothesis tests based on statistics used in existing attacks. The test-based approach has the advantage of being more efficient and more generalizable. This thesis applies both the attack-based approach and the test-based approach to better understand privacy risk for different applications. We use the attack-based approach to study privacy risk of machine learning models under specific adversarial models, and utilize the test-based approach to design generalizable privacy protection framework for releasing system logs.

To answer the second question, we focus on relaxing assumptions in existing privacy attacks on machine learning models. First, we study membership inference attacks and property inference attacks under a black-box adversary model that assumes the adversary can only interact with the model through its prediction APIs. Then, we propose more pragmatic adversary models by relaxing the assumptions on the adversary's knowledge about the training data distribution, the training model structure, and the attack target. By studying attack performances under these more pragmatic adversary models, we get a better understanding on the privacy risk in different real-life circumstances.

1.2 THREE LEVELS OF PRIVACY PROTECTIONS

Based on the understanding on privacy risk, we divide privacy protection into three different levels with distinct privacy-utility trade-offs.

Level 1: High Utility, Low Privacy Protection. At the first level, we consider data with high utility requirement and relatively low privacy protection. This type of data often share the following three properties. First, the data usually have high dimensionality and heterogeneous data structures. Second, the applications of the data are highly noise-sensitive. Third, although privacy concerns do exist, there is no single privacy requirement that fits all the applications. One example is the sharing of system logs. On the one hand, system contains sensitive information that may leak business secrets or security vulnerabilities. On the other hand, the availability of system logs from large companies is crucial for researchers to continually design new systems or improve the performance of existing ones. Therefore, there is an increasing need of sharing system logs under some privacy protection. Yet, strong theoretical privacy definition such as differential privacy does not fit the need of protecting business secrets and cannot achieve desirable level of data utility [9].

For this type of applications, we propose empirical protections to meet the strong utility requirement while offering some levels of privacy protection. We design various hypothesis tests to determine whether there is a risk for leaking the sensitive information and only perform obfuscation if a hypothesis test fails. This approach allows users to avoid unnecessary utility loss. If there is no strong indication that there is a privacy risk, the user can choose not to add any privacy protections and share the original data.

Level 2: High Privacy Protection, General Uses. At the second level, we consider sensitive data that need to be shared for general purposes. For example, datasets containing personal photos can be used in a wide range of applications including face recognition, human pose extraction, and mood detection. However, these photos are also extremely sensitive since they contain a lot of privacy information. To prevent the sensitive information from being leaked, it is important to ensure that the released data does not reveal identities of individuals in the dataset. Consequently, we want to ensure a strong privacy guarantee while preserving the distribution in the original dataset. To achieve these goals, we propose differentially private data generative framework G-PATE. Our approach leverages generative adversarial nets to generate data and exploits the Private Aggregation of Teacher Ensembles (PATE) framework to protect data privacy.

Level 3: High Privacy Protection, Specific Uses. At the third level, we consider sensitive data that are shared for specific applications. The privacy requirement for this type of data is two-fold. First, we need a strong theoretical privacy guarantee to protect private information. Second, we need to prevent the data to be used for unauthorized purposes. These goals can be achieved by sharing differentially private machine learning models trained

for the desired applications. Compared to training a model on differentially private synthetic data, ensuring privacy protection on the model often incurs less utility loss. Yet, this approach relies on differentially private machine learning algorithms for state-of-art models. In this thesis, we propose differentially private graph convolutional networks (DP-GCN), which is a powerful tool for graph analysis with strong privacy guarantee.

1.3 THESIS CONTRIBUTIONS AND ORGANIZATIONS

Based on the understanding of pragmatic privacy attacks on machine learning systems, this thesis proposes a multi-level privacy protection framework to provide different trade-offs between privacy protection and data utility.

The contribution of the thesis can be summarized as follows:

- We advance the understanding on privacy risks of machine learning models under more pragmatic adversary models.
- We propose a test-based approach to efficiently estimate the risk of various privacy attacks and identify potential privacy leakage.
- We design a privacy protection framework that combines data obfuscation techniques with privacy tests. This framework makes it possible to achieve empirical privacy protections under strong utility requirements.
- We propose a scalable differentially private data generative model to allow privacy-preserving data sharing for general purposes.
- We propose the first differentially private graph convolutional network based on node clustering.

The thesis is organized as follows. In Chapter 2, we present the related work in privacy of machine learning systems. In Chapter 3 to Chapter 5, we advance the understanding on privacy risk of machine learning systems with different attacks and measurements. Compared to existing attacks, our attacks consider more pragmatic adversarial models to better understand the risk under real-world attackers. In Chapter 6 to Chapter 8, we design three levels of protection mechanisms to achieve different trade-offs between privacy and utility. Specifically, in Chapter 6, we propose empirical protections for data with high utility requirement and relatively low privacy risk. In Chapter 7, we present a differentially private data generator to assist sharing sensitive data for general uses. In Chapter 8, we consider the sharing of

sensitive data for specific applications and design differentially private GCN models to help protect privacy in graph analysis. Chapter 9 concludes the thesis.

CHAPTER 2: RELATED WORK

In this chapter, we present the related work in privacy of machine learning systems.

2.1 PRIVACY ATTACKS ON MACHINE LEARNING MODELS

Membership Inference Attacks (MIA). In a membership inference attack, the adversary's goal is to infer the membership status of a target individual's data in the input dataset to some computation. For a survey, the adversary wishes to ascertain, from aggregate survey responses, whether the individual participated in the survey. For machine learning, the adversary wishes to ascertain whether the target's record was part of the dataset used to train a specific model. A successful MIA is a privacy violation because it indicates that the target individual is identifiable from the aggregated statistics or models.

One of the first prominent examples of MIA occur in the context of Genome-Wide Association Studies (GWAS). The seminal work of Homer et al. [10] show that p-values, a type of aggregated statistics routinely published when reporting the results of studies, could be used to successfully infer membership status. The experiment is performed on 86 individuals, all of which can be identified with little false positives. Although this attack requires that the adversary know the genome of the target individual, it teaches an important lesson: seemly harmless aggregate statistics may contain sufficient information for successful membership inferences, which leads to re-identification of individuals in the study. As a consequence of this attack, NIH removed all aggregate data of GWAS from public websites [11].

More recently, it was shown that machine learning models are vulnerable to black-box membership inference attacks. Shokri et al. [7] cast the attack into a classification problem and show that an attack classifier can infer record membership with a precision of 93.5%, when the target model is overfitted and has a testing accuracy around 65%. Hayes et al. [12] show that similar attacks are possible on generative target models, and Salem et al. [13] show that the attacker can also succeed only with access to data drawn from a different distribution and without knowledge of the target model structure.

Property Inference Attacks. Property inference attacks on ML models were first formulated by Ateniese et al. [14]. The work proposed a white-box attack based on a meta-classifier that takes model parameters as features and predicts whether the target model has the property P or not. The attack is shown to be effective on Support Vector Machine (SVM) and Hidden Markov Models (HMMs). Ganju et al. [8] extended the attack to Fully-Connected

Networks (FNNs) by improving the meta-classifier. Both attacks assume that the adversary has white-box access to the target model and can obtain data drawn from the same distribution as the training dataset.

Other Attacks on ML Models. Besides property inference attacks and membership inference attacks, ML models are shown to be vulnerable to a variety of attacks. Model inversion attacks [15, 16] infer the missing features based on the class label of a record. Model extraction attacks [17, 18, 19] infer the parameters or hyper-parameters of the target model based on its predictions on a set of queries. Adversarial attacks [20, 21, 22] trick ML models to give wrong predictions with high confidence by perturbing the queries. Poisoning attacks [23, 24, 25] inject malicious records to the training dataset to make ML models make wrong predictions.

2.2 DIFFERENTIAL PRIVACY AND RÉNYI DIFFERENTIAL PRIVACY.

Differential Privacy (DP). Proposed by Dwok et al., differential privacy [6] formalizes the vague concept of privacy into a provable property.

Definition 2.1 ((ε, δ)-Differential Privacy)**.** A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ε, δ)-differentially private if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for any neighboring datasets D and D':

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq \exp(\varepsilon) \Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{S}] + \delta. \tag{2.1}$$

DP guarantees privacy protection against an attacker with precise knowledge about the input dataset and *all* the entities in the universe except for the target individual. Follow-up works on DP try to relax the background knowledge by building more realistic background knowledge models [26, 27, 28, 29, 30]. Most of these relaxations can be unified under the framework of membership privacy [31], which shows that protecting private information is equivalent to preventing an attacker from knowing whether an individual is included in the input dataset. Specifically, DP is shown to be equivalent to membership privacy with mutually independent distributions. Other extensions on DP enhances protections on outliers in a data set while relaxes protections on the remaining examples [32, 33].

Rényi Differential Privacy. Rényi differential privacy is a natural relaxation of differential privacy. Defined below, its privacy guarantee is expressed in terms of Rényi divergence.

Definition 2.2 ((λ, ε)-RDP)**.** A randomized mechanism $\mathcal{M}$ is said to guarantee (λ, ε)-RDP with $\lambda > 1$ if for any neighboring datasets D and D',

$$D_\lambda \left(\mathcal{M}(D) \| \mathcal{M}\left(D'\right)\right) = \frac{1}{\lambda - 1} \log \mathbb{E}_{x \sim \mathcal{M}(D)} \left[\left(\frac{\mathbf{Pr}[\mathcal{M}(D) = x]}{\mathbf{Pr}\left[\mathcal{M}\left(D'\right) = x\right]} \right)^{\lambda - 1} \right] \leq \varepsilon. \tag{2.2}$$

(λ, ε)-RDP implies $(\varepsilon_\delta, \delta)$-differential privacy for any given probability $\delta > 0$.

Theorem 2.1 (From RDP to DP)**.** *If a mechanism $\mathcal{M}$ guarantees (λ, ε)-RDP, then $\mathcal{M}$ guarantees $(\varepsilon + \frac{\log 1/\delta}{\lambda - 1}, \delta)$-differential privacy for any $\delta \in (0, 1)$.*

Compared to DP, RDP supports easier composition of multiple queries and clearer privacy guarantee under Gaussian noise. Specifically, RDP could be easily composed by adding the privacy budget:

Theorem 2.2 (Composition of RDP)**.** *If a mechanism $\mathcal{M}$ consists of a sequence of $\mathcal{M}_1, \ldots, \mathcal{M}_k$ such that for any $i \in [k]$, $\mathcal{M}_i$ guarantees (λ, ε_i)-RDP, then $\mathcal{M}$ guarantees $(\lambda, \sum_{i=1}^k \varepsilon_i)$-RDP.*

Suppose f is a real-valued function, and the Gaussian mechanism is defined as follows:

$$\mathbf{G}_\sigma f(D) = f(D) + N\left(0, \sigma^2\right), \tag{2.3}$$

where $N\left(0, \sigma^2\right)$ is normally distributed random variable with standard deviation σ and mean 0. The Gaussian mechanism provides the following RDP guarantee:

Theorem 2.3 (RDP Guarantee for Gaussian Mechanism)**.** *If f has sensitivity 1, then the Gaussian mechanism $\mathbf{G}_\sigma f$ satisfies $(\lambda, \lambda / \left(2\sigma^2\right))$-RDP.*

Differentially Private Data Synthesis. Recently, various approaches have been proposed for differentially private data generation. Priview [34] generates synthetic data based on marginal distributions of the original dataset, and PrivBayes [35] trains a differentially private Bayesian network. However, these approaches are not suitable for image datasets since the statistics they use cannot well preserve the correlations between pixels in an image. Both DP-GAN [36] and PATE-GAN [37] apply differential privacy to the training process of generative adversarial networks (GAN). They both ensure differential privacy while training the discriminator, and the privacy property of the generator is guaranteed by the post processing property of differential privacy [38].

2.3 EMPIRICAL PRIVACY PROTECTION MECHANISMS

Log Anonymization. Prior studies have proposed various methods to hide user-identifiable information in logs. These studies can be classified into two categories: log anonymization and encryption. Log anonymization refers to the process of removing or redacting user identifiers in system logs. For example, there has been extensive research on IP anonymization [39, 40, 41, 42, 43, 44], ranging from truncating all IP addresses to prefix-preserving pseudonymization [43, 44]. Similarly, studies on timestamp anonymization has shown that it's possible to convert timestamps into relevant timestamps, which hide the exact time of the events but preserve the orders [45, 46]. On the other hand, some research has studied the use of encryption mechanisms in log privacy. Encryption methods, such as searchable encryption, can hide sensitive information in logs while allowing the administrator to do certain analysis [47, 48, 49, 50].

Data Obfuscation. Data obfuscation is a mechanism to protect privacy by adding misleading, false, or ambiguous information [51, 52]. For example, Sweeney [5] proposed the use of generalization and suppression to hide identifying information in a dataset containing person-specific records. Bakken et al [53] designed a set of obfuscation primitives and proposed properties to quantify the usefulness and privacy of the techniques. Some data obfuscation techniques such as generalization and suppression could be used to obfuscate logs.

Cryptographic Indistinguishability Obfuscation. Cryptographic indistinguishability obfuscation is a cryptographic primitive that provides a formal notion of program obfuscation. It requires that the obfuscation of two equivalent circuits C_0 and C_1 should be computationally indistinguishable [54]. This property can be guaranteed by algebraic hardness assumptions [54] or the security of other cryptographic primitives such as public-key functional encryption [55].

2.4 TRADE-OFFS BETWEEN PRIVACY AND UTILITY

There has been a long-lasting discussion around the trade-off between privacy and utility. In differential privacy, privacy and utility are distincted based on whether the information is about specific individuals or large populations [6]. A privacy budget ϵ is used to quantify the privacy cost and to balance the trade-off between utility and privacy. Following differential privacy, various privacy notions have been proposed to provide different ways to trade-off privacy for utility. For example, membership privacy [31] quantifies privacy risk as the adversary's ability to infer whether an entity is in the input dataset. It generalizes differential

privacy by supporting different distirbution families to instatiate the privacy definition, and differential privacy is equivalent to membership privacy under all mutually independent distributions. Blowfish privacy [56] extends differential privacy with a policy that specifies the information that must be kept secret and the constraints that may be known about the data. Compared to differential privacy and its extensions, this thesis balance the privacy-utility trade-off on a more general level. Specifically, we consider two questions: (1) what privacy protection criteria should be applied and (2) what data should be shared. Based on these two questions, we divide applications into three levels with different privacy and utility requirements and propose different solutions for each level.

CHAPTER 3: A PRAGMATIC APPROACH TO MEMBERSHIP INFERENCES ON MACHINE LEARNING MODELS

In this chapter, we revisit membership inference attacks from the perspective of a more sophisticated pragmatic adversary who carefully selects targets and make predictions conservatively. We design a new evaluation methodology that allows us to evaluate the membership privacy risk at the level of individuals and not only in aggregate. We experimentally demonstrate that highly vulnerable records exist even when the aggregate attack accuracy is close to 50% (baseline). This chapter is based on joint work with Lei Wang, Diyue Bu, Vincent Bindschaedler, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen [57].

Recent progress on machine learning has led to technological innovations for applications such as autonomous driving, face recognition, and natural language processing. But it has also uncovered new privacy threats. For example, in a Membership Inference Attack (MIA), an attacker queries a machine learning model in order to infer whether a specific target record was part of the training dataset.

Although seemly benign, inferring an individual's membership in a dataset or participation in a study can have serious privacy implications. For example, if the machine learning model was trained using medical records of patients suffering from a sensitive medical condition (e.g., cancer) then a successful membership inference may be devastating as it could reveal medical conditions an individual suffers from. A different way of conceptualizing membership inference is as a kind of re-identification attack from aggregated information (here the machine learning model). Viewed this way, as suggested in [58], protecting membership information is critical.

Recently, Shokri et al. [7] demonstrated the first MIA on classification models using only black-box access. This spurred further research into MIAs in a machine learning context [12, 13, 59]. In addition, some defensive measures have been proposed by Nasr et al. [60]. Despite this promising new research, there are always concerns about whether MIA indicate serious risks for widely used machine learning models. The concerns are two-fold. First, some prior MIAs require the adversary to have control of the training algorithm [61, 59], which may not always be realistic in practice. Second, despite recent work [7, 12, 13, 59] on MIA, it remains unclear what it means for MIAs to be successful and what is the actual privacy risk. For example, what does it mean for an adversary to achieve 80% accuracy in an MIA? What information does he learn about each individual in the dataset? What if the MIA achieves only 51.7% accuracy?

In this chapter, we make a step towards answering these questions by rethinking what

it means for a MIA to be successful from an adversary's point of view. We argue that the methodology used by prior work provides an incomplete picture. This can be understood as follows. For example, (in one case) prior work [7] reports an attack accuracy of 51.7% (whereas the random guessing baseline is 50%). This unequivocally demonstrates the existence of successful MIAs but does little to elucidate the actual privacy risk because it is compatible with two vastly different scenarios: (1) 1.7% of individuals having their membership status permanently and unequivocally at risk and the other 98.3% being safe; and (2) all individuals having a probability of 0.517 (instead of 0.5) of having their membership status correctly guessed (and anything in between these scenarios). As a privacy violation, the first scenario is arguably much more serious.

Prior work considers an *indiscriminate* adversary whose attack success is averaged over all targets, without regard to the cost of false positives or negatives. In contrast, we consider a *pragmatic* adversary who carefully selects targets based on their (perceived) vulnerability to membership inference and attempts to minimize false positives by trading off coverage for precision. We argue that such a conservative adversary is more realistic — and thus more reflective of the true privacy risk — because in the real world there is often a cost for making false accusations, thus making such an adversary value correct positive inferences more than correct negative inferences.

We propose novel attacks that better match this setting and allows us to distinguish between two critical aspects of membership inference: (a) the attacker's success averaged over targets, and (b) the attacker's success for a specific target averaged over the random idiosyncrasies of the model's training data and choices during training (e.g., initial random weights values). From a privacy perspective, both (a) and (b) should be minimized by prospective defenses. In fact, the only existing defense with provable privacy guarantee, i.e., differential privacy [62], puts a tight bound on both. In contrast, prior work has only focused on (a) thus providing an incomplete picture of the privacy risk. Indeed, our study reveals that a pragmatic adversary can achieve high precision (e.g., 95.05% on MNIST) in cases where prior work's methodology implies only barely above-the-baseline accuracy (i.e., 51.7%). It is worth noting that such findings occur even when the machine learning model is not overfitted, a setting for which prior work on black-box MIA reports significantly lower risk of membership privacy violation.

3.1 ADVERSARY MODEL

We consider an adversary mounting a MIA against already trained machine learning models. We assume that the adversary has black-box access to the target models, i.e., he can issue

arbitrary queries and retrieve the answers (e.g., the probability vector) from the models; the number of queries, however, may be limited.

In this section, we formulate the attack model used in prior MIAs as the indiscriminate attack and propose our more sophisticated attack model named pragmatic attack.

3.1.1 Indiscriminate Attack

In an *indiscriminate attack*, an attacker performs the attack over a set of randomly picked records, and the attack advantage is evaluated over *all* the records. Let D be a set of records, and $\mathcal{A}$ be the training algorithm. The attack could be described as the following distinguishing game between a user and an adversary:

1. The user randomly splits D into a training set S_{train} and a testing set S_{test} of the same size.
2. The user trains a model $M = \mathcal{A}(S_{\text{train}})$. The adversary has black-box access to M.
3. $\forall r \in D$, $x_r = 1$ if $r \in S_{\text{train}}$, otherwise $x_r = 0$.
4. $\forall r \in D$, the adversary obtains a guess $x'_r \in \{0, 1\}$.
5. $\forall r \in D$, the adversary succeeds if $x'_r = x_r$, otherwise the adversary fails.

The adversary's probability of success p is calculated as the number of successes of the adversary divided by the number of records in D. Prior work on MIA have used two metrics to evaluate the performance of an indiscriminate attack: the attack accuracy and the adversary advantage. In most attacks [7, 13], the attack performance is evaluated by the attack accuracy, which equals to the probability of success p. In addition, Yeom et al. [59] defined the adversary advantage as the probability of winning the distinguishing game over random guessing, and the advantage is calculated as $2p - 1$.

3.1.2 Pragmatic Attack

Although the indiscriminate attack model has been widely adopted in prior attacks and defenses, it ignores the potential influence of the cost of attacks and false positives. In this chapter, we consider a *pragmatic attack*, where the adversary carefully selects attack targets and tries to minimize false positives. Let D be a set of records, and $\mathcal{A}$ be the training algorithm. We formalize the attack process as the following distinguishing game:

1. The adversary chooses a target $r \in D$.
2. The user randomly splits D into a training set S_{train} and a testing set S_{test} of the same size.
3. The user trains a model $M = \mathcal{A}(S_{\text{train}})$. The adversary has black-box access to M.
4. $x_r = 1$ if $r \in S_{\text{train}}$, otherwise $x_r = 0$.
5. The adversary produces a guess $x'_r \in \{1, \perp\}$ and performs an attack only if $x'_r = 1$.
6. If $x'_r = 1$ and $x_r = 1$, the adversary succeeds. If $x'_r = 1$ and $x_r = 0$, the adversary fails.

We repeat steps (2)-(6) to estimate the adversary's probability of success on the target record r over the randomness of sampled training set S_{train} and the training algorithm $\mathcal{A}$.

Pragmatic attacks are different from indiscriminate attacks in two aspects. First, instead of naively attacking all the records, a pragmatic adversary carefully selects the attack targets to avoid wasting time and resources on records that are unlikely to be vulnerable to membership inferences. The process of target record selection greatly reduces the chance of making false predictions and increases the probability of success. Second, a pragmatic adversary tries to minimize false positives because there is often a high cost for making false accusations. In a pragmatic attack, an adversary makes a positive inference (i.e., $x'_r = 1$) only if she has high confidence that the target record is in the training dataset, otherwise she makes no inferences (i.e., $x'_r = \perp$).

We define two metrics to evaluate the performance of the attack: (1) the *precision* of the attack is the probability of success among all the positive inferences (i.e. $\Pr[x_r = 1 \mid x'_r = 1]$); (2) the *coverage* of the attack is the probability of making a positive inference when the target record is in the training dataset (i.e. $\Pr[x'_r = 1 \mid x_r = 1]$). We evaluate the attack precision and coverage of each target record over the randomness of the training algorithm and sampling of training dataset.

The adversary makes a false positive inference when $x'_r = 1$ and $x_r = 0$. False positives are often associated with high cost and could reduce the adversary's credibility, so a pragmatic adversary attempts to minimize the number of false positives and maximize the attack precision. On the contrary, the adversary makes no inferences when $x'_r = \perp$, so a low coverage does not incur extra cost for the adversary. Therefore, it is acceptable to have a relatively low attack coverage.

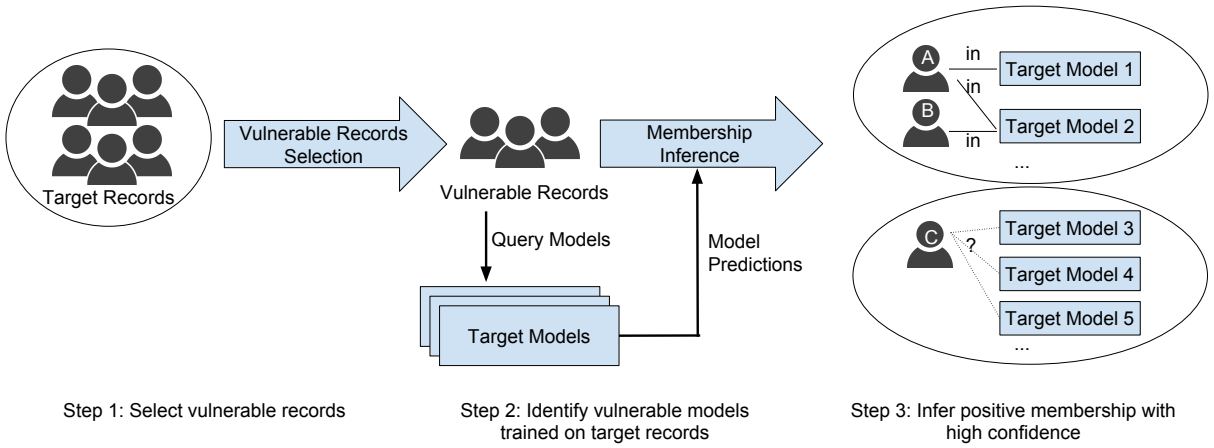


Figure 3.1: Attack Overview.

3.1.3 Adversary Knowledge

Similar as the previous work [7], we further assume that the adversary either (1) knows the structure of the target model (e.g., the depth and the number of neurons each layer of the neural network) and the training algorithm used to build the model, or (2) has black-box access to the machine learning algorithm used to train the model. We also assume that the adversary has some prior knowledge about the population from which the training records are drawn. Specifically, the adversary can access a set of records that are drawn independently from that population, which may or may not overlap with the actual training data for the target models; but the adversary does not have any additional information about whether these records are present in the training data. These records can often be obtained from public dataset with similar attributes or from previous data breaches.

3.2 PRAGMATIC MEMBERSHIP INFERENCE ATTACK

3.2.1 Attack Overview

The goals of our attack are different from the goals of an indiscriminate attack in two aspects: (1) the adversary is only interested in positive membership inferences because positive membership information is more valuable to the adversary and more risky to the users. Positive membership inference allows the adversary to associate public available information (i.e., the machine learning models) with some identifiable auxiliary information (i.e., the record of an individual known to the adversary). Because this association is similar to re-identifying individuals in an anonymized dataset, positive membership inference attack

has been considered as a type of re-identification attack in prior work [58]. Moreover, given a correct positive membership inference, the adversary knows that the individual is a participant of a study, which may further leak more information about that individual. (2) The adversary wants to re-identify individuals in the training dataset *with high precision* because false inferences can be costly. Around these goals, we design a three-step pragmatic attack as shown in Fig. 3.1. Below, we briefly explain each step of the attack.

Step 1: Selecting Vulnerable Target Records In an overfitted model, almost all records are vulnerable to MIA, so a indiscriminate attack can achieve high accuracy. However, when the model is well-generalized, the model gives similar predictions to members and non-members of the training dataset. Therefore, identifying vulnerable target records is the key to an effective pragmatic attack. First, we select vulnerable records by estimating the number of neighbors they have in the the sample space represented by the records available to the adversary. Records with fewer neighbors are more vulnerable under MIA because they are more likely to impose unique influence on the machine learning models. In order to identify neighbors of a given record, we train reference models to imitate the behavior of target models. We further construct a new feature vector for each record based on the intermediate outputs of reference models on this record, which implies this record's influence on the target machine learning model.

Step 2: Identifying Vulnerable Models Next, we query the target models and identify the models that are trained on target records. Specifically, we design two attack methods distinguished by their queries to the target models: A direct inference attack infers the membership of a target record based on the model's prediction on that record; an indirect inference attack infers the membership of a target record based on the record's influence on the model's predictions on seemingly uncorrelated records (called *enhancing records*). We use novel techniques that iteratively search for and select enhancing records. Our indirect inferences using the enhancing records can successfully infer the presence of a target record without querying it. Moreover, the indirect inferences sometimes outperform direct inferences by accumulating more information from multiple queries. Note that although we design and evaluate our attack with multiple target records and target models, in practice, the adversary may choose a single target model or a single target record to attack.

Step 3: Inferring Positive Membership Finally, we make positive membership inference over the combinations of all target records and target models. Since there is often a high cost when making incorrect inference, we only infer a target record to be in the target model

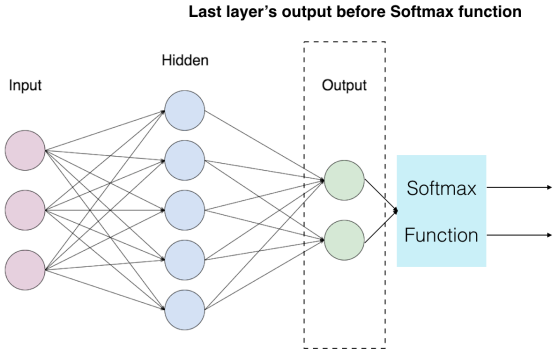


Figure 3.2: Last layer output of a two-layer neural network. We use the last layer output of locally trained nueral networks as features for vulnerable record selection.

if the predictions of the model indicate a high probability of success in the attack. We use hypothesis testing methods to make the decision: under the null hypothesis the record is not present in the training dataset; under the alternative hypothesis the target record is in the training dataset. We reject the null hypothesis when the p-value is smaller than a cut-off threshold.

3.2.2 Building Reference Models

We exploit a target record's unique influence on the outputs of a machine learning model to infer the presence of the record in the training set of the target model (called target training set). To identify such influence, we need to estimate the model's behavior when the target record is *not* in the target training set. To achieve this goal, we build *reference models*, which are trained using the same algorithm on *reference datasets* sampled from the same space as the target training set, but not containing the target record. The process of building reference models are illustrated below.

To start with, we need to construct k reference datasets with the same size as the target training set. Since most practical machine learning models are trained on large training datasets, it is difficult for an adversary to get access to an even larger dataset with k times records as the target training set. Consequently, if we build the reference datasets by sampling without replacement from the whole set of reference records, the resulting datasets may share many records, and the reference models built from them would be alike and give similar outputs. To address this issue, we use bootstrap sampling [63] to generate the reference datasets, where each dataset is sampled with replacement. Bootstrap sampling reduces overlaps among reference datasets, providing a better approximation of datasets sampled

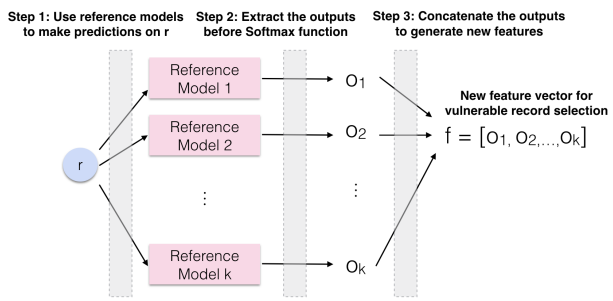


Figure 3.3: Features for vulnerable records selection. We concatenate intermediate outputs of locally trained reference models and use them as features for vulnerable records selection.

from distribution of the target training set. Each reference dataset is then used to train a reference model using the same training algorithms as used for training the target model.

3.2.3 Selecting Vulnerable Records

Not all training records are vulnerable to MIA. In general, we want to measure the potential influence of a target record so as to select vulnerable records with the greatest influences and subject them to MIA in the subsequent steps. It is worth noting that, although the training records imposing unique influence on the model are often *outlier records* (i.e., with distinct feature vectors) in the training set, the outlier records do not always have unique influence on the model because the training algorithm may decide that some features should be given higher weights than others and some features should be combined in the model. For example, a neural network trained on hand written digit datasets learns the contour of written digits is more important feature than individual pixels [64]. Therefore, instead of using the input features, we extract high level features more relevant to the classification task to detect vulnerable records.

Specifically, when attacking neural networks (e.g., see Figure 3.2 for a two-layer fully connected neural network), we construct new feature vectors by concatenating the outputs of the last layer before the Softmax function from the reference models (Figure 3.3), as the deeper layers in the network are more correlated with the classification output [65]. We then measure the unique influences of each record using its new feature vector. Let $\mathbf{f}$ be the the new feature vector of the record r. We call two records r_1 and r_2 *neighbors* if the cosine distance between their feature vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ is smaller than a *neighbor-threshold* α.

Note that the neighboring records are difficult to be distinguished by MIA because they have similar influence on the model. When a neighbor of r occurs in the training dataset,

the model may behave as if r is used to train the model, leading to the incorrect membership inference result. Our goal is to select the vulnerable records in the entire record space with fewer or no neighbors likely to be present in the training set (assuming the training records are independently drawn from the record space) as putative targets of MIA.

Given a training dataset with N records and a reference dataset with N' records, both sampled from the same record space, and a target record r, we count the number of neighbors of r in the reference dataset, denoted as N'_n. Then, the expected number of neighbors of r in the training dataset, N_n, can be estimated as $\mathbb{E}[N_n] = N'_n \times \frac{N'}{N}$.

A record r is considered to be potentially vulnerable (and as the attack object), only if $\mathbb{E}[N_n] < \beta$, where β is the *probability-threshold* for target record selection. We stress that the approach for vulnerable records selection presented here relies only on the record space (represented by the reference records accessible by an adversary) and the reference models (built using reference records), and is independent of the target model; as a result, the computation can be done off-line even when used to attack a machine learning as a service (MLaaS).

3.2.4 Direct Inference

In a pragmatic attack, the goal of the adversary is to achieve high precision on the selected target records instead of achieving high accuracy over all records. Therefore, we attack each target record separately by computing the deviation between its output given by the target model and those given by the reference models. We expect that each training record has a unique influence on the model, which can be measured by comparing the target model's output with the output of reference models (trained without the target record) on the record. We quantify the difference between the outputs using the log loss function. Given a classifier M and a record r with class label y_r, let p_{y_r} be M's output probability of class label y_r. The log loss function [66] $\mathcal{L}(M, r)$ is defined as:

$$\mathcal{L}(M, r) = -\log p_{y_r}. \tag{3.1}$$

The log loss function is commonly used as a criterion function [66] when training neural network models. $\mathcal{L}(M, r)$ is small when M gives high probabilities on correct labels.

Given a target model M, a target record r, and k reference models, we first obtain the log loss of all the reference models on r as $L_1, L_2, \ldots, L_k$. We view these losses as samples independently drawn from a distribution $\mathcal{D}(L)$, and estimate the empirical cumulative distribution function (CDF) of $\mathcal{D}_L$ as $F(L)$, which takes a real-valued loss L as input. We

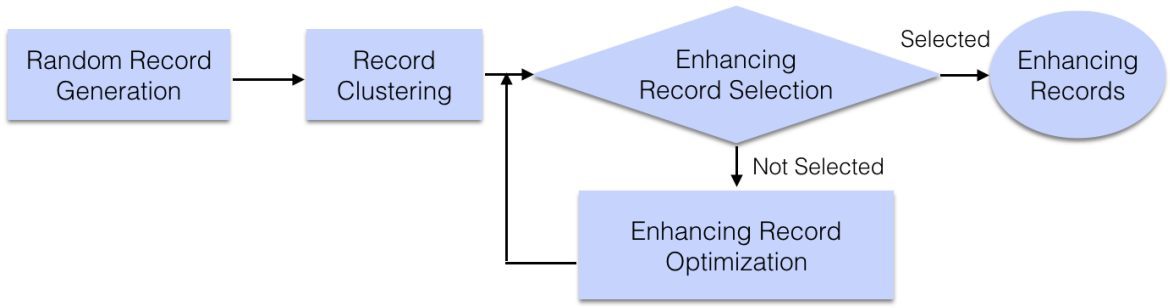


Figure 3.4: Steps for generating enhancing records.

use the shape-preserving piecewise cubic interpolation [67] to smooth the estimated CDF. Based on the log loss of the target model M on the target record r, $\mathcal{L}(M, r)$, we estimate the confidence of r to be present in the training set by performing a left-tailed hypothesis test: under the null hypothesis H_0, r is not present in the training set (i.e., $\mathcal{L}(M, r)$ is randomly drawn from $\mathcal{D}(L)$), while under the alternative hypothesis r is used to train M (i.e., $\mathcal{L}(M, r)$ is smaller than samples in $\mathcal{D}(L)$ because of the influence of r in the training). Therefore, we calculate the p-value as:

$$p = F(\mathcal{L}(M, r)), \tag{3.2}$$

which gives the confidence that r is used for training M only if p is smaller than a threshold (e.g. 0.01) so that the null hypothesis is rejected.

3.2.5 Indirect Inference

Besides reducing a model's loss on its own, a training record also influences the model's outputs on other records. This influence is desirable to improve model generalization: in order to give correct predictions on unseen records, a model needs to use the correlation it learns from a training record to make predictions on queries with similar features. On the other hand, however, these influences can be exploited by an adversary to obtain more information about the target record through multiple queries to enhance MIA. Interestingly, we show that MIA can be achieved by queries of records seemingly uncorrelated with the target record, making the attack hard to detect and defense.

The key challenge for inference without querying the target record is to efficiently identify the *enhancing records* whose outputs from the target model are expected to be influenced by the target record. To address this problem, we develop a method consisting of the following

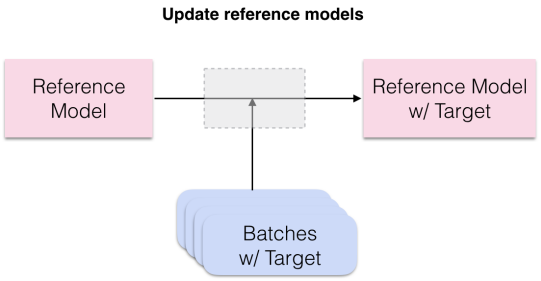


Figure 3.5: Building positive reference models by updating the model with the training set including the reference records plus the target record.

steps: random record generation, record clustering, enhancing record selection, and enhancing records optimization (as shown in Figure 3.4).

Random Record Generation To start with, we randomly generate records from which the enhancing records are selected. Specifically, we adopt one of the following two methods for random record generation: (1) when the feature space is relatively small, we uniformly sample records from the whole feature space; (2) when the feature space is large, since the chance of getting enhancing records by uniform sampling is slim, we generate random records by adding noise to pre-selected vulnerable target records. We use Gaussian noise for numerical attributes and candle noise [68] for categorical attributes.

Enhancing Record Selection To identify records whose target model's output may be influenced by the target record r, we approximate the target model's behavior using a group of *positive reference models* that are trained using reference records plus the target record r. To save the effort of retraining the positive reference models, we add the target record into batches sampled from the original reference dataset and update the reference models by training on the batches plus the target record. Figure 3.5 shows the process of updating reference models.

We select the *enhancing records* by comparing the predictions between the positive reference models (i.e., "in models") and the original reference models (that are trained without the target records, i.e., "out models"). We denote the ith original and the ith positive reference model as M_{ref_i} and $M^r_{\text{ref}_i}$, respectively. Given a record r with class label y_r and another arbitrary record q, let $M(q, y_r)$ be the model M's output probability of y_r on the query q.

We calculate r's influence on q as follows:

$$I\left(r,q\right)=\frac{1}{k}\sum_{i=1}^{k}\mathrm{t}\left(M_{\mathrm{ref}_i}^{r}\left(q,y_r\right)-M_{\mathrm{ref}_i}\left(q,y_r\right)\right)\ , \tag{3.3}$$

where k is the total number of original (or positive) reference models, and t is a threshold function defined as follows:

$$\mathrm{t}\left(x\right)=\begin{cases}1 & \text{if}\quad x>0,\\ 0 & \text{otherwise.}\end{cases} \tag{3.4}$$

Algorithm 3.1 Enhancing Records Selection Algorithm

1: $I\left(r,q\right)\leftarrow\sum_{i=1}^{k}\mathrm{t}\left(M_{\mathrm{ref}_i}^{r}\left(q,y_r\right)-M_{\mathrm{ref}_i}\left(q,y_r\right)\right)/k$
2: **if** $I>\theta$ **then**
3: Accept q
4: **else**
5: Reject q
6: **end if**

We identify a randomly generated record q as an enhancing record for the record r if $I(r,q)$ approaches 1, which indicates that adding r to the training dataset *almost always* increase the models' output probability on the class label y_r for the query q. In practice, we use q in the MIA on the target record r only if $I(r,q)$ is greater than a threshold θ (e.g. 0.95). Algorithm 3.1 summarizes the entire algorithm for query selection.

Enhancing Record Optimization When the target model has a large record space (e.g., with high-dimension feature vectors), the chance of finding an enhancing record among randomly generated records is slim. To address this issue, we propose an algorithm to search for enhancing records for a target record r by optimizing the following objective function:

$$\max_{q} I\left(r,q\right), \tag{3.5}$$

where $I\left(r,q\right)$ is the influence function defined in Equation 3.3. Optimizing $I\left(r,q\right)$ is time-consuming because $I\left(r,q\right)$ consists of a non-differentiable threshold function t. Therefore, instead of solving the optimization function in equation 3.5, For simplification, we approximate the maximization of $I\left(r,q\right)$ with the minimization of the sum of multiple hinge loss functions

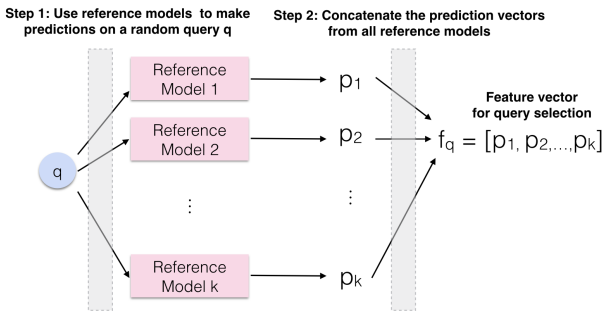


Figure 3.6: Generating query features for query selection.

defined as follows [69]:

$$\min_{q} \sum_{i=1}^{k} \max\left(0, \gamma - \left(M^{r}_{\mathrm{ref}_i}\left(q, y_r\right) - M_{\mathrm{ref}_i}\left(q, y_r\right)\right)\right), \tag{3.6}$$

where γ is a parameter indicating the margin width. If a randomly generated record are rejected by the query selection algorithm, we minimize the objective function in Equation 3.6 using gradient descent [70] to check if the resulting record is acceptable as an enhancing record.

Record Clustering (Optional) Note that it is inefficient to repeat the query selection and optimization algorithms on all random records because the predictions of the models on most records are highly correlated: the models giving high output probabilities on some record are also likely to give high output probabilities on correlated records. To improve the efficiency of query selection, we propose an algorithm to identify the *least correlated* enhancing records from a large number of randomly generated records.

First, we estimate the correlation between records based on the model's predictions on them. We construct a feature vector $\mathbf{f}_q$ for a record q by concatenating the reference models' outputs on it (Figure 3.6). If two queries q_1 and q_2 have highly correlated feature vectors, the models' outputs on q_2 do not add much information to the models' outputs on q_1.

Next, we formulate the problem of selecting a subset of least correlated records as a graph theoretical problem. We build a graph where records are the nodes and pairwise correlation between records is the weight on edges connecting the corresponding nodes. This allows us to recast our problem as the k-lightest subgraph problem [71], which is NP-hard. We obtain an approximate solution using hierarchical clustering [72]. For this, we cluster the records into k disjoint clusters based on their pairwise cosine distance. Finally, in each cluster, we

select the record with least average cosine distance to all other records in the same cluster.

As shown in Figure 3.4, we use the enhancing record clustering algorithm before the enhancing record selection and enhancing record optimization steps to improve the efficiency of the attack.

Indirect Inference with Multiple Queries After identifying multiple enhancing records, we repeat the attack in section 3.2.4 by querying each of these records. Because the outputs on these queries may be correlated, we combine the resulting p-values using Kost's method [73], with the covariance matrix estimated from the query features generated in the query selection step (Figure 3.6).

3.3 EVALUATION

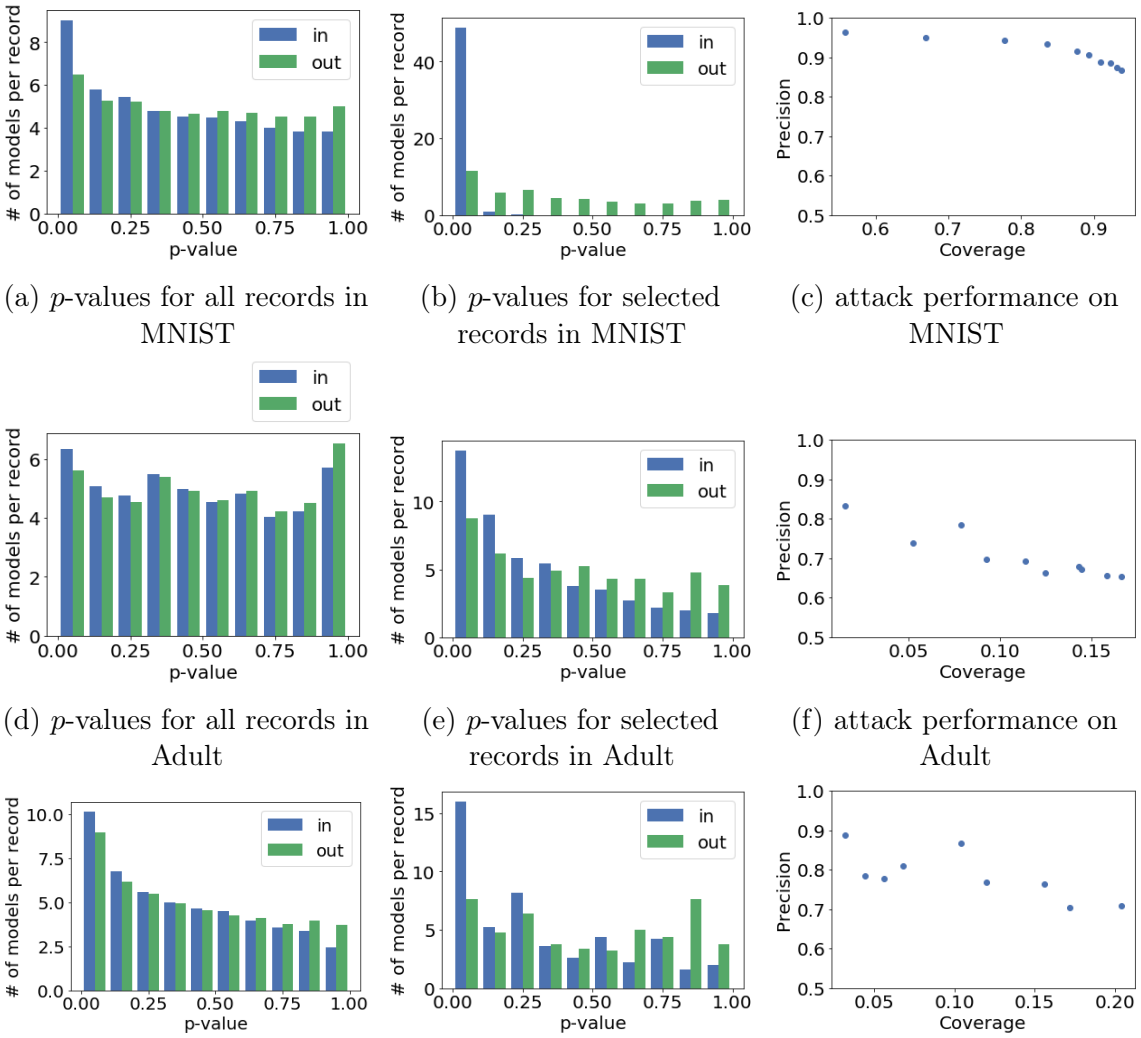
3.3.1 Experimental Setup

We evaluated the performance of our attack from the following three aspects: the precision of the attack, the coverage of the attack, and the effectiveness of vulnerable record selection method.

For each dataset, We constructed 100 target models. To get a better understanding of MIA's performance, we wanted the baseline precision to be 0.5 for each target record. That is, each target record should occur in 50 out of the 100 target models. Therefore, we generated training datasets by randomly splitting the target records into two datasets of the same size, each serving as a training set for a target model. We repeated this process for 50 times and generated the training datasets for 100 target models.

The *precision* of the attack is the percentage of successful inferences (i.e., the target record is indeed in the training dataset) among all inferences. The *coverage* of the attack is the percentage of successful inferences among all the cases that the target record is in the training set (i.e. 50 times the number of records). In practice, a high precision is often more important than a high coverage because there is usually a high cost associated with making false inferences.

We define *true positive (TP)* to be the case that the target record is indeed in the training dataset when the adversary inferred it as in and *false positive (FP)* to be the case that the target record is *not* in the training dataset when the adversary inferred it as in. We evaluate the effectiveness of our vulnerable record selection method by looking at the true positives and false positives of each vulnerable record under different selection criteria.



(a) p-values for all records in MNIST

(b) p-values for selected records in MNIST

(c) attack performance on MNIST

(d) p-values for all records in Adult

(e) p-values for selected records in Adult

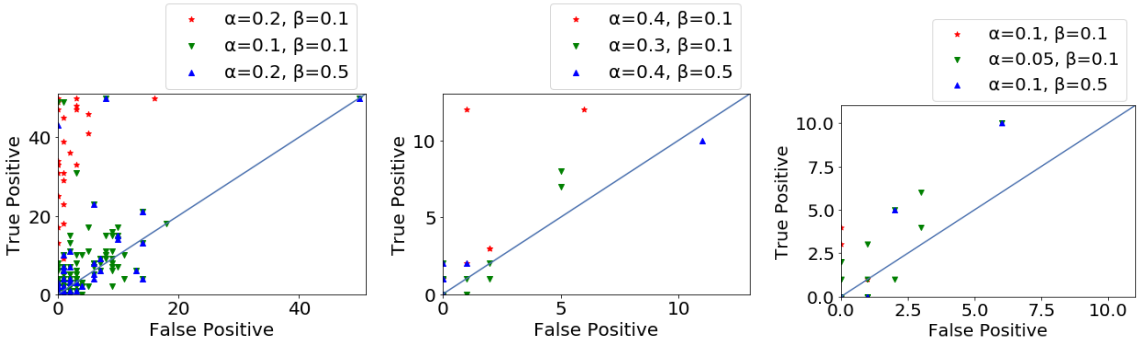
(f) attack performance on Adult

(g) p-values for all records in Cancer

(h) p-values for selected records in Cancer

(i) attack performance on Cancer

Figure 3.7: Evaluation of MIA on Adult dataset. In (a), (d), and (g), we performed hypothesis testing on *all* records over 100 target models. There was no significant difference between the distributions of p-values for models trained with the target record (labeled "in") and models trained without the target record (labeled "out"). This result indicates that the attack cannot achieve high precision without the step of vulnerable record selection. In (b), (e), and (h), we performed the same hypothesis testing on the selected vulnerable records over 100 target models. Our attack was effective because there was a distinction between the p-value distributions of "in" models and "out" models. Most models with a small p-value were models trained with the target record (i.e. "in" models). (c), (f), and (i) show the varying precision and coverage with cut-off p-value ranging from 0.005 to 0.05. Our attack focused on achieving high attack precision because false accusations can be costly for attackers.



(a) MNIST Dataset (b) Adult Dataset (c) Cancer Dataset

Figure 3.8: Effectiveness of vulnerable record selection in a pragmatic attack. We evaluated the effectiveness of vulnerable record selection by plotting the number of true positive inferences and false positive inferences of each selected record. Each point in the figure represents a target record. Points can overlap because the attack can have the same performance on different records. For each target record, the attack was performed over 100 target models (50 "in" models and 50 "out" models). Records selected under different criteria were plotted with different colors and shapes. Records with cosine distance smaller than α were considered as neighbors.

3.3.2 Dataset

UCI Adult The UCI Adult dataset [74] is a census dataset containing 48,842 records and 14 attributes. The attributes are demographic features and the classification task is to predict whether an individual's salary is above \$50K a year. We normalized the numerical attributes in the dataset and used one hop encoding [75] to construct the binary representation of categorical features. We randomly selected 20,000 records for training target models, and each training dataset contains 10,000 records. The remaining 28,842 records served as the adversary's background knowledge.

UCI Cancer The UCI cancer dataset [74] contains 699 records and 10 numerical features ranging between 1 to 10. The features are characteristics of the cell in an image of a fine needle aspirate (FNA) of a breast mass. The classification task is to determine whether the cell is malignant or benign. We randomly selected 200 records for training, and each training dataset contains 100 records. The remaining 499 records served as the adversary's background knowledge.

MNIST Dataset The MNIST dataset [76] is an image dataset of handwritten digits with 60,000 handwritten training examples and 10,000 testing examples. The images are normalized such that the digits are positioned at the center of the 28x28 pixel field. The

classification task is to predict which digit is represented in an image. We randomly selected 20,000 images for training and 40,000 images as the adversary's background knowledge. Each training set for target models and reference models contains 10,000 images. We used the 10,000 testing images to calculate testing accuracy.

3.3.3 Models

Neural Network For the Adult dataset, we constructed a fully connected neural network with 2 hidden layers with 10 units and 5 units respectively. We use `Tanh` as the activation function and `SoftMax` as the output layer. The model is trained with batchsize of 100 and 20,000 epochs. For the MNIST dataset, we constructed 2 convolutional layers with ReLu as the activation function, followed with max pooling layers. We then added a fully connected layer of 1,024 neurons, and we also used dropout techniques to reduce overfitting. Finally, we added an output layer and a Softmax layer. The model is trained with batchsize of 50 and 10,000 epochs. For the Cancer dataset, we used a vanilla neural network with no hidden layer. The model is trained with batchsize of 10 and 3,000 epochs.

Google ML Engine Since the Google Predictions API used in the prior attack is deprecated, we used Google ML Engine to train target models on ML cloud. When training the model, we used the sample code provided by Google, which has pre-built model structures for training models on Adult dataset and MNIST dataset. Specifically, for Adult dataset, the sample code uses Google estimator [77] which hides low-level model structure from the user; for MNIST dataset, the sample code builds a neural network with 2 fully-connected hidden layers.

3.3.4 Direct Inference

We evaluated the performance of direct inferences by their precision and coverage on different datasets and models. We set a fixed vulnerable record selection criterion for each dataset. The neighbor threshold α was 0.2, 0.4, and 0.1 for MNIST, Adult, and Cancer respectively. This threshold represented the maximum cosine similarity between neighbors. Records with cosine similarity smaller than α were considered as neighbors. Therefore, this threshold varied for different datasets depending on the dimensionality of records. We evaluated the influence of this threshold later in this section. We selected records with probability threshold $\beta = 0.1$. That is, the likelihood that a neighbor of the record occurs in the training dataset was smaller than 0.1.

Dataset (Model)	Vulnerable Records	Precision	Coverage
MNIST	27	95.05%	66.89%
Adult	13	73.91%	5.23%
Cancer	5	88.89%	3.20%
MNIST (Google)	1	100%	4%
Adult (Google)	7	80%	2.67%

Table 3.1: Performance of Direct Inference. We measured the performance of a direct inference attack by its precision and coverage. To achieve a high precision, we selected a few vulnerable records (neighbor threshold $\alpha = 0.2$ for MNIST, 0.4 for Adult, and 0.1 for Cancer; probability threshold $\beta = 0.1$), and made positive inferences only when attack confidence is high ($p \leq 0.01$).

Dataset (Model)	Training Accuracy	Testing Accuracy
Adult	0.85 ± 0.01	0.85
Cancer	0.95 ± 0.04	0.94 ± 0.03
MNIST	0.99	0.98
Adult (Google)	0.84 ± 0.03	0.84 ± 0.02
MNIST (Google)	0.90	0.90

Table 3.2: Training and Testing Accuracy of Target Models. All the target models were well-generalized models with difference between training and testing accuracy smaller than 0.01.

In Fig. 3.7 we plotted the average number of models per record with different attack p-values. The models trained with the target records are labeled as "in" and the models trained without the target records are labeled as "out". The attack was effective only when there was a distinction between the p-value distribution of "in" models and "out" models. The figure shows the necessity of selecting vulnerable records before doing the inferences. When the attack hypothesis testing was performed on *all* target records, the p-value distributions of "in" models and "out" models were indistinguishable. Therefore, the attack was unlikely to have a high precision no matter what p-value cutoff we selected. On the other hand, when we performed the same hypothesis testing on the selected vulnerable records, there was a clear distinction between the p-value distributions of "in" models and "out" models, which led to successful membership inference attacks.

The cut-off p threshold controls the trade-off between precision and coverage. We would only make a positive inference if the p-value obtained from the attack is smaller than the threshold. Since there is a cost for false inferences, we chose small p thresholds in the attack. In Fig. 3.7, we plotted the different attack precision and coverage obtained by cut-off p

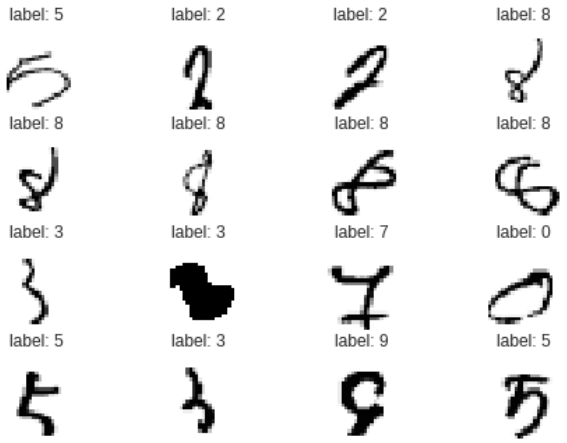


Figure 3.9: Vulnerable Examples in MNIST Dataset

threshold varying between 0.005 to 0.05.

Our attack mechanism was less effective on the Adult Google ML model since we did not have access to the exact model structure due to the use of Google estimator. Instead, we used raw features to select target records. This limitation reduced the number of vulnerable target records we identified from 13 to 7. However, it also showed that the attack is possible even when the adversary does not know the model structure.

3.3.5 Selection of Vulnerable Records

The p-value distributions in Fig. 3.7 shows the importance of vulnerable records selection. We further explored how this step influenced the attack performance by changing the neighbor threshold α and the probability threshold β. Fig. 3.8 shows the records selected by different selection thresholds. Each point in the figure represents a target record. Points at the upper left corner are more vulnerable to MIA than those near the baseline. Smaller neighbor thresholds or higher probability thresholds increased the number of selected vulnerable target records. However, as we tried to attack more records at the same time, there was a higher chance that we would make false positive inferences due to the influence of a record similar to one of the target records, which decreased the attack precision.

To study what kinds of records are vulnerable, we plotted the vulnerable target records selected from MNIST dataset with $\alpha = 0.2$ and $\beta = 0.1$ (Figure 3.9). As we expected, some of the vulnerable target records are outliers in the dataset. However, some vulnerable examples actually increase model utility by providing rare but useful features for the classification task. For example, the images of digit 8 written in different directions may help a model on

Dataset	Cut-off p-value	Prec. (direct)	Coverage (direct)	Prec. (indirect)	Coverage (indirect)
Adult	0.01	-	0	1	14%
	0.1	70.83%	34%	75%	24%
Cancer	0.01	1	6%	-	0
	0.1	66.67%	52%	88.89%	16%
MNIST	0.01	96.15%	1	1	2%
	0.1	89.29%	1	52.38%	22%

Table 3.3: Comparison between direct and indirect inferences. We performed the attack on the same selected record with direct inference and indirect inference. The result indicates that membership inference attack is feasible without directly querying the target record. On Adult dataset, indirect inferences even outperformed direct inferences.

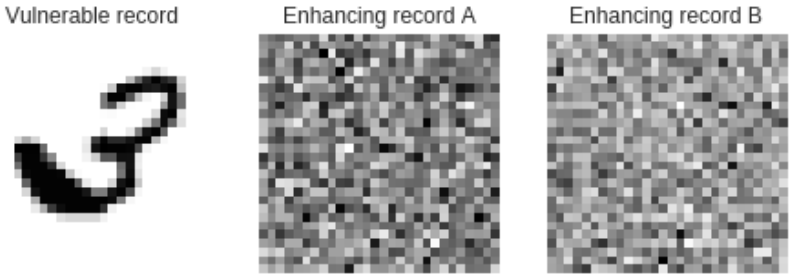


Figure 3.10: A vulnerable record from MNIST with its two enhancing records. In practice, it is difficult to find out what the target record is, by looking at the enhancing records used by an adversary.

recognizing similar written digits in testing examples. However, since these images are rare in the dataset, they have a unique influence on the target models, making them vulnerable to our attack, and the fact that this influence is useful in predicting unseen examples does not mitigate the risk.

3.3.6 Indirect Inference

For some vulnerable target records, we achieved the same level of attack performance by querying enhancing records. For each dataset, we randomly sampled $5,000$ records, selected 50 of them by record clustering, and tested them with the enhancing record selection algorithm. If less than 10 enhancing records were selected, we ran the enhancing record optimization algorithm to improve the records. The initial records for the Cancer dataset and the Adult dataset were randomly sampled from the feature space while the records for the MNIST dataset were generated by adding noise to the target records due to the large

Regularization Coefficient λ	**Training Acc.**	**Test Acc.**	**# of Target Records**	**Prec.**	**Coverage**
0	0.99	0.98	52	90.84%	68.31%
0.001	0.99	0.99	1	1	54.8%
0.01	0.98	0.98	1	93.36%	4%

Table 3.4: Attack Performance w.r.t. Regularization ($\alpha = 0.2$, $\beta = 2$, $p \leq 0.01$). We applied L2 regularization with varying coefficients λ. Experiment results show that applying regularization reduced, but did fully eliminate the privacy risk of a pragmatic adversary.

feature space.

We selected 1 target record in each dataset. For the Cancer dataset, we selected 47 enhancing records whose euclidean distance to the target record range between 6 and 19.3 with a selection criterion $I(r, q) > 0.95$. Since the Cancer dataset has relatively low dimensional features, enough enhancing records were accepted, and enhancing record optimization was not needed. For the Adult dataset, we relaxed the enhancing record selection criterion to $I(r, q) > 0.9$ and found 15 enhancing records after the optimization step. For the MNIST dataset, we further relaxed the enhancing record criterion to $I(r, q) > 0.8$ due to the high dimensional feature space. We identified 41 enhancing records generated by adding noise to the target record.

Table 3.3 shows the performance of indirect inferences. For both the Cancer dataset and the Adult dataset, attacking with the enhancing records has compatible performance as querying the target record. Moreover, for the Adult dataset, querying the target record did not successfully infer any cases with a 0.01 cut-off p-value, but by combining the predictions on enhancing records, we achieved a precision of 1 and a coverage of 14%. For the MNIST dataset, we achieved a precision of 1 and a coverage of 2% when $p \leq 0.01$. Although this performance is less impressive compared to a direct inference on the same record (whose precision and recall are both close to 1) it's still an indication that membership inference attack can succeed without querying the target record.

The effectiveness of indirect inferences shows that prior defenses [78, 79] based on direct inferences could *not* eliminate the risk of membership inferences. Moreover, we plotted both the target record and the enhancing records and found that the enhancing records in no means represent the target record, indicating that our attack is hard to detect (Figure 3.10).

3.3.7 Influence of Regularization

Regularization is a common method for improving model generalization. It is shown to be an effective defense against the prior MIA [7]. To study its effectiveness on our attack, we applied L2 regularization on neural networks trained on MNIST set even though the models were *not overfitted*. In doing so, we limited the model capacity which increased the risk of underfitting. Specifically, when the regularization coefficient λ went from 0.001 to 0.01, testing accuracy decreased by 0.01 indicating that the model might be underfitted due to over regularization.

Table 3.4 shows the model accuracy and attack performance before and after applying L2 regularization with varying coefficients λ. Applying regularization reduced the number of vulnerable target records in the dataset, but did not completely eliminate the privacy risk. The remaining vulnerable records were attacked with high precision. Specifically, when L2 regularization was applied with coefficient $\lambda = 0.01$, we still identified 1 vulnerable target record, which was inferred with precision close to 1.

Applying regularization mitigated the model's privacy risk of *some* vulnerable individuals but did not eliminate the risk of *all* individuals. Moreover, since the most vulnerable record was identified with high precision, regularization may not be a good approach when the data owner wants to provide privacy protection for *all* individuals whose records are in the dataset.

3.3.8 Comparison with Indiscriminative Attacks

To compare with the attack proposed by Shokri et al. [7], we reproduced the attack on the same target models and the same vulnerable records in our attack. Specifically, we trained one attack classifier per class for each dataset. The attack classifiers are neural networks with one hidden layer of 64 units. We used `ReLU` as the activation function and `SoftMax` as the output layer. We only performed the attack when the probability given by the attack classifier was higher than a certain threshold (called attack confidence threshold). We evaluated the performance of the attack under various attack threshold as shown in Table 3.5. The attack precision was relatively low (e.g. $< 70\%$) on all three datasets even when a high attack confidence threshold was used.

Dataset	Attack Confidence Threshold	Precision	Coverage
Cancer (3 records)	0.8	50.25%	40%
	0.9	-	0
Adult (13 records)	0.6	66.67%	4.92%
	0.7	-	0
MNIST (27 records)	0.6	50%	56.25%
	0.7	19.6%	6.25%
	0.8	-	0

Table 3.5: Performance of the attack of Shokri et al. [7] on the same target models and the same target records. To imitate the attack strategy of a pragmatic adversary, we performed prior attack on the selected target records and made predictions only when the attack classifier has high confidence. However, the prior indiscriminative attack could not achieve high precision even under a low coverage.

3.4 DISCUSSION

In this section, we explain the limitations of our attack, and further discuss the potential mitigation to the information leaks in machine learning models.

3.4.1 Limitations

In the meantime, our current attack is preliminary. Our techniques for identifying outliers cannot find all vulnerable instances: it is possible that some instances not considered to be outliers by our current design still exert unique influences on the model, which need to be better understood in the follow-up research. Moreover, the current way to search for the for the enhancing records, through filtering out random queries, is inefficient, and often does not produce any results. More effective solutions could utilize a targeted search based upon a better understanding about the relations between the target record and other records. Fundamentally, it remains unclear how much information about the training set is leaked out through querying a machine learning model and whether more sensitive techniques can be developed to capture even a small signals for a record's unique impact.

3.4.2 Mitigation

Adversarial Regularization and Adversarial Examples Prior research has used adversarial regularization [78] and adversarial examples [79] as defenses against membership inference attacks. However, both defenses assume an indiscriminate adversary that trains an

attack classifier based on the model's *direct* prediction on the target records. Meanwhile, the pragmatic attack strategy in our work is more sophisticated, and our indirect queries are hard to identify. Therefore, it is challenging to model the attacker for privacy regularization or to generate adversarial examples for the attack.

Generalization and perturbation As mentioned earlier, generalization has limited effect on mitigating our more sophisticated membership inference attack: as demonstrated in our study, even after applying the L2 regularization (with a coefficient of 0.01), still a vulnerable record in MNIST dataset can be attacked with a precision of 1 (Section 3.3.7). In the meantime, adding noise to the training set or to the model to achieve differential privacy can suppress the information leak [62]. However, in the presence of high-dimensional data, which is particularly vulnerable to our attack, perturbation significantly undermines the utility of the model before its privacy risk can be effectively controlled [80]. So we believe that a practical solution should apply generalization and perturbation together with proper training set selection, detecting and removing those vulnerable training instances.

Training record selection We believe that there is a fundamental contention between selecting useful training instances, which bring in additional information, and suppressing their unique influence to protect their privacy. An important step we could take here is to automatically identify outliers and drop those not contributing much to the utility of the model. To this end, new techniques need to be developed to balance the risk mitigation and the utility reduction for those risky instances. A machine learning model could be built to automatically decide whether an instance should be in the training set or not.

3.5 CONCLUSIONS

In this chapter, we take a step forward to better understanding information leaks from machine learning models. In contrast to prior work, we consider a more sophisticated pragmatic adversary who carefully selects targets and makes predictions conservatively. We demonstrate new membership inference attacks allowing such an adversary to identify vulnerable targets, and we deploy a novel methodology to evaluate the risk. Our results show that this new methodology better reflects the privacy risk of membership inference. In fact, it highlights cases where prior work underestimates the risk, achieving low attack accuracy (barely above the random-guessing baseline), whereas our pragmatic adversary still achieves high precision (at the cost of lower coverage). Specifically, our study reveals that a pragmatic adversary can achieve high precision (e.g., 95.05% on MNIST) in cases where prior work's

methodology implies only barely above-the-baseline accuracy (i.e., 51.7%). In addition, our study highlights the conflict between selecting informative training instances and preventing their identification through their unique influences on the model, and points to the direction of using training data analysis and selection to complement existing approaches.

CHAPTER 4: TOWARDS MEASURING MEMBERSHIP PRIVACY

In this chapter, we investigate and analyze membership attacks to understand why and how they succeed. Based on this understanding, we propose Differential Training Privacy (DTP), an empirical metric to estimate the privacy risk of publishing a classifier when methods such as differential privacy cannot be applied. DTP is a measure of a classifier with respect to its training dataset, and we show that calculating DTP is efficient in many practical cases. We empirically validate DTP using state-of-the-art machine learning models such as neural networks trained on real-world datasets. Our results show that DTP is highly predictive of the success of membership attacks and therefore reducing DTP also reduces the privacy risk. This chapter is based on joint work with Vincent Bindschaedler and Carl A. Gunter [81].

Machine learning models are widely used to extract useful information from large datasets and support many popular Internet services. Companies like Amazon [82], Google [83], and Microsoft [84] have started to provide Machine Learning-as-a-Service (MLaaS). Data owners upload their data and obtain black-box access to a classification model which can be queried through an API.

Recent academic work has pointed out several security and privacy issues with this MLaaS paradigm. Models can be stolen or reverse engineered [85], sensitive population-level information can be inferred [86, 87]; even the corresponding training datasets can be targeted by inference attacks. In particular, Shokri et al. [7] propose a membership attack to infer sensitive information about individuals whose data records are part of the training dataset.

Membership attacks are not new or specific to MLaaS and are known credible threats in various contexts such as in Genome-Wide Association Studies (GWAS) [10]. In principle, membership attacks are easily thwarted by ensuring that the model is trained using a differentially private process. Unfortunately, it is not always feasible to use a learning algorithm that satisfies differential privacy. Some classification models cannot readily be trained in this way, or doing so may come at the cost of an unacceptable utility loss.

This issue is exacerbated by the fact that if a classification model is trained without differential privacy, then little is known about its membership privacy risk; there is a gap in our ability to analyze the risk. At the same time, there is no reason to believe that all classification algorithms leak the exact same amount about their training datasets — a model that is badly overfitted has the potential to leak more than one which is not. Yet there is currently no framework or principled way to measure this in practice.

In this chapter, we investigate why and how membership inference attacks succeed. We

derive a general attack framework and perform experiments on state-of-the-art classifiers trained on real-world datasets. Our goal is to design a metric which reflects membership privacy risk and can easily be calculated on a classifier.

We identify a simple measure called *Differential Training Privacy (DTP)* which quantifies the risk of membership inference of a record with respect to a classifier and its training data; the higher the DTP value, the higher the risk. We extend DTP to a metric over the classifier, by computing the DTP value of all records in the training dataset and taking the maximum—the *worst case* risk. DTP is not a substitute for a differentially private learning algorithm. Rather DTP provides an objective basis for decision making. For example, when two classifiers exhibit similar performance, it is preferable to publish the one with the lowest DTP.

Informally, given a classifier $\mathcal{A}(T)$ trained on a dataset T, the membership leakage of a record $t \in T$ is quantified by comparing that classifier's predictions to those of a classifier trained without record t, i.e., $\mathcal{A}(T \setminus \{t\})$. We assume black-box access, so an adversary can only learn information by querying the classifier. In this setting, membership attacks are predicated on distinguishing whether the classifier was trained on T or $T \setminus \{t\}$. This is what DTP measures. Differences in predictions can occur for any query, but we initially focus on *direct attacks* which expect the maximum difference to be observed when querying features of t.

We provide experimental validation of direct attacks on both traditional classifiers such as naive Bayes and logistic regression and state-of-the-art models such as neural networks trained on two real-world datasets: a purchase dataset containing the shopping history of 300,000 individuals, and the popular UCI Adult dataset. Specifically, we perform several membership inference attacks on these classifiers, including the most effective attack known. Results suggest that DTP is a powerful predictor of the accuracy of direct attacks. Concretely, for neural networks learned on the purchase dataset, the Pearson correlation coefficient of the maximum membership attack accuracy with DTP is 0.8936. For classifiers with DTP-values under 0.5, none of the attacks we performed ever inferred membership status of any individual with accuracy greater than 66.5% (baseline: 50%). By comparison our attacks almost always have over 90% accuracy when DTP is larger than 4.

Although we do not know of any practical *indirect attacks*—which query the classifier for features other than those of t—we cannot exclude the possibility that such attacks may outperform direct ones. In fact, we produce a counter-example that this is indeed possible. We explore whether this situation occurs for known classification algorithms and derive a set of theoretical results, including a simple criterion (for classifiers) called *training stability*. For algorithms which satisfy training stability, the direct attack is always superior to *any*

indirect attack.

Contributions. We propose Differential Training Privacy (DTP) to quantify membership inference risk of publishing a classifier. DTP is used to inform the decision of whether to release a classifier when techniques to achieve differential privacy cannot be employed. We advocate for the DTP-1 hypothesis: if a classifier has a DTP value above 1, it should not be published. We test this hypothesis by designing effective attacks on records with DTP greater than 1 based on different classifiers and datasets.

We present a general membership attack framework and evaluate three types of attacks on several classifiers trained on two real-world datasets, including the most effective attack known prior to this work—which we improve upon. Our empirical study of the relationship between the accuracy of membership attacks and DTP, reveals the latter to be a powerful predictor of the former.

We establish training stability as an important desideratum for classifiers, and prove that naive Bayes, random decision trees, and linear statistical queries satisfy it but k-NN does not.

4.1 PROBLEM STATEMENT

Consider a data owner with a dataset D. We assume that the dataset is divided into multiple classes and each record in the dataset consists of a class label and a vector of features. We also assume that the data owner randomly partitions the dataset into a training set and a validation set. A machine learning model is trained on the training set. The model captures the correlations between the features and the class labels. It takes a feature vector as input and outputs a vector of probabilities for each class.

Suppose the data owner wants to make the model available for public queries. That is, he intends to allow anyone to submit a feature vector and get predictions from the model. In this chapter, we want to estimate the membership inference risk of releasing the model based on simple measurements and theoretical analysis. Specifically, given a machine learning model, we want to answer the following questions: *Is there a privacy risk if the model is open to public queries? Are certain models riskier than others? Which records have higher privacy risks?*

We consider the privacy risk in the setting of membership privacy. The privacy risk of a record is estimated by the adversary's ability to infer whether the record is a part of the training set. We estimate this risk under the assumption that the machine learning models are trained by trusted parties. That is, we expect the parties training and releasing the model

have the goal of protecting the training set to the extent this is practical and will therefore not be motivated to create a covert channel by embedding private information into the model's predictions. Under this assumption, we inspect the risk of accidental privacy leakage—the risk that a machine learning algoirthm would accidentally learn too much information about an individual record during the training process.

To estimate privacy risk under a strong adversary, we assume that the adversary knows all the records in D, the size of the training set sampled from D, and the machine learning algorithm used to train the model. We assume the training set to be uniformly sampled from D. Therefore, each record in D is equally likely to be included in the training set, and the adversary does not know which records are included. The goal of the adversary is to perform a membership inference attack by querying the machine learning model. That is, given a particular target record $t \in D$, he wants to infer whether t is used to train the model he queries.

Notations. Let T be the training set of the model. Since T is randomly sampled from D, we call D the *candidate set* for T.

We define X^m to be a set of all possible features and Y to be the set of all possible class labels: $\{y_1, y_2, \ldots, y_k\}$. Each record $\mathbf{z} \in D$ can be divided into two parts: the feature vector $\mathbf{x} \in X^m$ and the class label $y \in Y$. Let $\mathcal{A}$ be the classification algorithm and $c = \mathcal{A}(T)$ be the output classifier. We assume that for each query $\mathbf{x}$, $c(\mathbf{x})$ returns a vector of conditional probability of all class labels $y \in \{y_1, y_2, \ldots, y_k\}$ given feature vector $\mathbf{x}$. We use $p_c(y \mid \mathbf{x})$ to represent the conditional probability of class label y given feature $\mathbf{x}$, predicted by classifier c. That is, $c(\mathbf{x}) = (\mathrm{p}_c(y_1 \mid \mathbf{x}), \mathrm{p}_c(y_2 \mid \mathbf{x}), \ldots, \mathrm{p}_c(y_k \mid \mathbf{x}))$. For classifiers that do not directly provide predicted probabilities, these can be obtained through normalization over the class labels.

In membership inferences, the adversary wants to infer whether a specific record $t = (\mathbf{x}^{(t)}, y^{(t)}) \in D$ is part of the training set T. We call t the *target record* of the attack.

There are two approaches that an adversary can take to perform a membership inference attack on a target record t. He can launch a *direct attack* by querying the features of the target record $\mathbf{x}^{(t)}$. Or, he can perform an *indirect attack* by querying some feature vector $\mathbf{x} \neq \mathbf{x}^{(t)}$. Intuitively, a direct attack should have better performance than an indirect attack because querying the features of t should give more information about t compared to querying other features. In this chapter, we first study the membership privacy risk under this assumption. In section 4.4, we test this assumption by analyzing some commonly used classifiers.

4.2 DIFFERENTIAL TRAINING PRIVACY

We propose a measure of membership privacy risk of a target record t with respect to classification algorithm $\mathcal{A}$ and training set T.

Definition 4.1 (Differential Training Privacy)**.** A record $t \in T$ is *ϵ-differentially training private* (ϵ-DTP) with respect to classification algorithm $\mathcal{A}$ and training set T, if for all $\mathbf{x} \in X^m$ and $y \in Y$, we have

$$p_{\mathcal{A}(T)}(y \mid \mathbf{x}) \leq e^{\epsilon} p_{\mathcal{A}(T\setminus\{t\})}(y \mid \mathbf{x}) \tag{4.1}$$

and

$$p_{\mathcal{A}(T)}(y \mid \mathbf{x}) \geq e^{-\epsilon} p_{\mathcal{A}(T\setminus\{t\})}(y \mid \mathbf{x}). \tag{4.2}$$

That is, the target record t is DTP with algorithm $\mathcal{A}$ and training set T if the predicted conditional probability of any class label y given any feature vector $\mathbf{x}$ does not change much when t is removed from T. By definition, a record t with low DTP only has a small influence on the output of the classifier $\mathcal{A}(T)$. Since a classifier's output is also influenced by other factors such as random initialization and unexpected records in the training set, from the adversary's perspective, the small influence by t is indistinguishable from the influence by other uncertain factors. Therefore, records with low DTP are less vulnerable to membership inference attacks.

Unlike differential privacy [62], DTP is a local property related to the training set. Therefore, DTP is experimentally measurable given a target record t, a training set T, and a classification algorithm $\mathcal{A}$. We use the following definition of DTP metric to quantify the privacy risk of the target record t.

Definition 4.2. (DTP Metric). Given classification algorithm $\mathcal{A}$, training set T, and target record t, the *differential training privacy metric* $DTP_{\mathcal{A},T}(t)$ is the least ϵ such that t is ϵ-DTP with $\mathcal{A}$ and T.

In practice, $\mathrm{DTP}_{\mathcal{A},T}(t)$ is calculated as the maximum ratio between the predictions given by $\mathcal{A}(T)$ and $\mathcal{A}(T \setminus \{t\})$ for all $\mathbf{x} \in X^m$ and for all $y \in Y$. That is,

$$\mathrm{DTP}_{\mathcal{A},T}(t) = \max_{\mathbf{x}\in X^m, y\in Y} \epsilon_t^{(\mathbf{x},y)}, \tag{4.3}$$

where

$$\epsilon_t^{(\mathbf{x},y)} = \max\left(\frac{p_{\mathcal{A}(T)}(y \mid \mathbf{x})}{p_{\mathcal{A}(T\setminus\{t\})}(y \mid \mathbf{x})}, \frac{p_{\mathcal{A}(T\setminus\{t\})}(y \mid \mathbf{x})}{p_{\mathcal{A}(T)}(y \mid \mathbf{x})}\right). \tag{4.4}$$

$\mathrm{DTP}_{\mathcal{A},T}(t)$ can be naively measured by brute force over all $\mathbf{x} \in X^m$ and all $y \in Y$. However, considering the potentially large size of X^m, this approach is neither practical nor efficient. Therefore, we propose pointwise differential training privacy (PDTP) as a relaxation of DTP.

Definition 4.3. (Pointwise Differential Training Privacy). A record $t \in T$ is *ϵ-pointwise differentially training private* (ϵ-PDTP) with respect to classification algorithm $\mathcal{A}$ and training dataset T, if for all $y \in Y$, we have

$$p_{\mathcal{A}(T)}(y \mid \mathbf{x}^{(t)}) \leq e^{\epsilon} p_{\mathcal{A}(T \setminus \{t\})}(y \mid \mathbf{x}^{(t)}) \tag{4.5}$$

and

$$p_{\mathcal{A}(T)}(y \mid \mathbf{x}^{(t)}) \geq e^{-\epsilon} p_{A(T \setminus \{t\})}(y \mid \mathbf{x}^{(t)}). \tag{4.6}$$

Similarly we propose the following definition for the metric $\mathrm{PDTP}(\mathcal{A}, T)$:

Definition 4.4. (PDTP Metric). Given classification algorithm $\mathcal{A}$, training set T, and target record $t \in T$, the *pointwise differential training privacy metric* $PDTP_{\mathcal{A},T}(t)$ is the least ϵ such that t is ϵ-PDTP with $\mathcal{A}$ and T.

PDTP is a relaxation of DTP which bounds the change of the classifier's response on a single query $\mathbf{x}^{(t)}$, when t is removed from the training set. Because of this, PDTP can be efficiently calculated given any classification algorithm $\mathcal{A}$, training set T, and target record t by training an alternative classifier $\mathcal{A}(T \setminus \{t\})$. This process is similar to the leave-one-out (LOO) cross-validation technique used in machine learning [88]. Since LOO is a core technique for evaluating a machine learning model, there is considerable experience with both learning algorithms for which its calculation is easier and optimizations to improve this performance.

The measurement $\mathrm{PDTP}_{\mathcal{A},T}(t)$ is useful for two different reasons: First, $\mathrm{PDTP}_{\mathcal{A},T}(t)$ is a lower bound of $\mathrm{DTP}_{\mathcal{A},T}(t)$. When $\mathrm{DTP}_{\mathcal{A},T}(t)$ cannot be efficiently calculated, data owners can use $\mathrm{PDTP}_{\mathcal{A},T}(t)$ as an optimistic estimation of a classifier's privacy risk. If a target record t has high PDTP with $\mathcal{A}$ and T, releasing the classifier $\mathcal{A}(T)$ brings high privacy risks for t. Therefore, PDTP can be used as an indicator of membership privacy risk. Second, PDTP reflects the performance of a direct membership attack. When the adversary uses $c(\mathbf{x}^{(t)})$ to infer the membership of t, it is sufficient to bound the change of $c(\mathbf{x}^{(t)})$ when t is removed from the training set of c. Since we assume direct membership attacks have better performance than indirect ones, $\mathrm{PDTP}_{\mathcal{A},T}(t)$ is a good estimation of membership privacy risk of t.

4.3 CASE STUDIES

DTP measures the sensitivity of a target record t on the predictions of a classifier $\mathcal{A}(T)$. Intuitively, the larger t's influence on the predictions of $\mathcal{A}(T)$ is, the more these predictions leak about t. This, in turn, makes t more vulnerable to membership inference attacks. However, to use DTP to calculate the membership risk, we still need to answer the following: *How do we use PDTP to estimate the risk of membership attacks? How accurate are these estimations? What values of PDTP indicate a potential privacy risk?*

In this section, we answer these three questions through a series of experiments on direct membership attacks.

To demonstrate PDTP's effectiveness in measuring risks of membership attacks, we study the performance of three types of direct membership attacks on different datasets and classification algorithms. We find that, when a membership inference attack is effective, i.e., the attack accuracy is greater than 0.7, $\text{PDTP}_{\mathcal{A},T}(t)$ is highly correlated with the attack's accuracy on t, and the correlation is higher for attacks with higher accuracy.

To identify high-risk records, we use the *DTP*-1 *hypothesis*: if a classifier has a DTP value above 1, it should not be published. Since PDTP is a lower bound for DTP, we use PDTP measurements to identify records that do not satisfy DTP-1 criterion and demonstrate effective membership attacks on these records.

4.3.1 Datasets

We first introduce datasets used in the experiments.

UCI Adult Dataset (Adult). The dataset [89] contains 48,842 records extracted from the 1994 Census Database. Each record has 14 attributes with demographic information such as age, gender, and education. The class attribute is the income class of the individual: > 50K or $<= 50$K. The classification task is to predict an individual's income class based on his demographic information. We use all the features except the final weight (fnlwgt) attribute, which is a weight on the Current Population Survey (CPS) file used for accurate populations estimates. We randomly sample 2,000 records as candidate set D, and 1,000 records out of D as training set.

Purchase Dataset (Purchase). Similar to the purchase dataset in [7], we construct a dataset containing user's purchase history based on Kaggle's "acquire valued shoppers" challenge. The original contains the user's transaction histories, including product category, product brand, purchase quantity, purchase amount, etc. We pre-process the dataset by constructing one record for each customer. We use the product category attribute in the

original dataset to create 836 binary feature attributes. Each feature attribute is a product category (e.g., sparkling water), and the value of the attribute is true if and only if the corresponding customer has purchased this product in the past year. We cluster the dataset into 10 clusters using k-means based on Weka's implementation [90]. Each cluster represents a type of consumer buying behavior. We use the cluster index of each record as its class label. The classification task is to predict a consumer's buying behavior based on products he has purchased. We randomly sample 2,000 records as candidate set D, and 1,000 records out of D as training set.

4.3.2 Machine Learning Models

We study the performance of membership attack and PDTP measurements on three different machine learning models.

Neural Networks (NN). We build a fully connected neural network with one hidden layer of 64 units and a `LogSoftMax` layer. We use `Tanh` as the activation function and negative log-likelihood criterion as the classification criterion. We use a learning rate of 0.01 for both datasets. The maximum epoch of training is set to 100 for the adult dataset and 30 for the purchase dataset. When preprocessing the adult dataset, we convert categorical attributes into binary attributes and normalize all the numerical attributes.

Naive Bayes Classifiers (NB). We build a Naive Bayes classifier [91] to predict the class label based on Bayes Theorem under the assumption of conditional independence. We use Laplace smoothing [92] to smooth the categorical attributes in the dataset.

Logistic Regressions (LR). We build a logistic regression model using Weka's implementation of Logistic model trees [93].

Binning the Predictions. We limit the precision of the model outputs using data binning technique with a bin size of 0.01. Since the outputs of the classifiers are probabilities in the range of $[0, 1]$, we divide this range into 100 bins of the same size. Instead of returning the original output of a classifier, we make the model return the center of the bin to which the original output belongs. For example, if a classifier predicts a class label to have 0 probability given a feature vector, the output of the classifier would be 0.005, which is the center of the first bin. This binning technique prevents models from leaking private information that does not significantly contribute to their accuracy. It also prevents PDTP measurements from getting unreasonably large due to close-to-zero denominators.

4.3.3 Attacks

Given a target classifier $c = \mathcal{A}(T)$ and a target record t, a membership attack distinguishes between the following hypotheses:

$$H_0 : \quad t \notin T \qquad \text{and} \qquad H_1 : \quad t \in T \ .$$

In all of the following attacks, we assume that the adversary gets the target classifier's prediction on the target record $c(\mathbf{x}^{(t)}) = \big(\mathrm{p}_c(y_1 \mid \mathbf{x}^{(t)}), \mathrm{p}_c(y_2 \mid \mathbf{x}^{(t)}), \ldots, \mathrm{p}_c(y_k \mid \mathbf{x}^{(t)})\big)$, and tries to launch a membership attack on t using this information. We also assume the adversary is powerful enough to know the size of training set T and has access to the candidate dataset D and the classification algorithm $\mathcal{A}$ based on Kerckhoffs's principle.

Let $q_i = \mathrm{p}_c(y_i \mid \mathbf{x}^{(t)})$ $(1 \leq i \leq k)$. Here, q_i represents the class probability for class label y_i predicted by the target classifier given the features of the target record. Therefore, the vector $\mathbf{q} = c(\mathbf{x}^{(t)}) = (q_1, q_2, \ldots, q_k)$ can be viewed as a probability distribution over all the possible class labels.

Untargeted Attacks. We reproduce the membership attack of [7] on a neural network model learned on the purchase dataset. This attack converts the membership inference problem into a classification task with two class labels: class label "in" represents hypothesis H_1 $(t \in T)$, and class label "out" represents hypothesis H_0 $(t \notin T)$. Concretely, the attack consists of two steps:

Step 1: Training Shadow Classiffiers. The adversary trains shadow classifiers to simulate the behavior of the target classifier $\mathcal{A}(T)$. First, he samples M shadow datasets $T_1, T_2, \ldots, T_M$ of the same size as the target dataset T. Then, he trains M shadow classifiers $\mathcal{A}(T_1), \mathcal{A}(T_2), \ldots, \mathcal{A}(T_M)$ using the same classification algorithm as the target classifier $\mathcal{A}(T)$. In experiments, we take $M = 20$.

Step 2: Building the Attack Classifier. The adversary uses the shadow classifiers to label each record in the candidate dataset D according to Algorithm 4.1. The algorithm takes the shadow classifiers and the candidate dataset as input and outputs a dataset D_{attack}, which serves as the training set for the attack classifier.

In experiments, the attack classifier is a fully connected neural network with one hidden layer of 32 units and a `LogSoftMax` layer. We use `ReLU` as activation function and negative log-likelihood criterion as classification criterion. We set the learning rate to 0.01 and the maximum epochs of training to 30 iterations.

Step 3: Launching the Attack. Given a target record t, the adversary constructs a new feacture vector by concatenating the original feature vector $\mathbf{x}^{(t)}$, the original class label $y^{(t)}$,

Algorithm 4.1 Step 2 of Untargeted Attack

Require: A set of shadow classifiers $\{\mathcal{A}(T_1), \mathcal{A}(T_2), \ldots, \mathcal{A}(T_M)\}$, candidate set D
Ensure: Training set of the attack classifier D_{attack}

```
D_attack ← ∅
for j = 1, 2, ..., M do
  for r ∈ D do
    q^(r) ← (p_{A(T_j)}(y_1 | x^(r)), ..., p_{A(T_j)}(y_k | x^(r)))
    f^(r) ← (x^(r), y^(r), q^(r))
    if r ∈ T_j then
      D_attack ← D_attack ⋃ {(f^(r), in)}
    else
      D_attack ← D_attack ⋃ {(f^(r), out)}
    end if
  end for
end for
```

and the target model's prediction on the target record $c(\mathbf{x}^{(t)})$. That is,

$$\mathbf{f}_{\text{attack}}^{(t)} = \left(\mathbf{x}^{(t)}, y^{(t)}, c(\mathbf{x}^{(t)})\right). \tag{4.7}$$

The adversary queries the attack classifier with the new feature vector and gets a prediction consisting of two probabilities: p_{in} is the probability of class label "in", and p_{out} is the probability of class label "out". The advesary accepts hypothesis H_1 if, and only if, $p_{\text{in}} > p_{\text{out}}$.

We call this type of attack an *untargeted attack* because the attack classifier obtained from step 2 can be used to attack all the records in the candidate dataset. Therefore, when the adversary tries to attack multiple records, he only needs to run step 1 and step 2 once. Step 3 of the attack, which needs to be repeated on each targeted record, has much lower computational overhead. Although this attack is more efficient when the adversary wants to find out *any* vulnerable records, it has lower accuracy compared to some of the targeted attacks.

Distance-Based Targeted Attacks. When the adversary has a specific target record in mind, he can design attacks tuned to perform well only on the target record. We call this type of attacks *targeted attacks*. In a distance-based targeted attack, an adversary uses shadow models to estimate the average predictions of classifiers satisfying hypothesis H_0, and those of classifiers satisfying hypothesis H_1. Then he calculates the distance between $c(\mathbf{x})$ and the two average predictions and accept the hypothesis under which the average predictions are closer to $c(\mathbf{x})$. Concretely:

Step 1: Training Shadow Classifiers. Let $n = |T|$. The adversary uniformly samples M

datasets $T'_1, T'_2, \ldots, T'_M$ of size $n-1$ from $D \setminus \{t\}$, and takes $T_j = T'_j \bigcup \{t\}$ for all $1 \le j \le M$. The adversary trains a pair of shadow classifiers $\mathcal{A}(T_j), \mathcal{A}(T'_j)$ for all $1 \le j \le M$, and gets their predictions on the target record:

$$\mathbf{p}_j^{\text{in}} = \left(\mathrm{p}_{\mathcal{A}(T_j)}(y_1 \mid \mathbf{x}), \mathrm{p}_{\mathcal{A}(T_j)}(y_2 \mid \mathbf{x}), \ldots, \mathrm{p}_{\mathcal{A}(T_j)}(y_k \mid \mathbf{x})\right), \tag{4.8}$$

and

$$\mathbf{p}_j^{\text{out}} = \left(\mathrm{p}_{\mathcal{A}(T'_j)}(y_1 \mid \mathbf{x}), \mathrm{p}_{\mathcal{A}(T'_j)}(y_2 \mid \mathbf{x}), \ldots, \mathrm{p}_{\mathcal{A}(T'_j)}(y_k \mid \mathbf{x})\right).$$

Take $\overline{\mathbf{p}}_{\text{in}} = \frac{1}{M}\sum_{j=1}^{M} \mathbf{p}_j^{\text{in}}$ and $\overline{\mathbf{p}}_{\text{out}} = \frac{1}{M}\sum_{j=1}^{M} \mathbf{p}_j^{\text{out}}$. Like the query result $\mathbf{q}$, $\overline{\mathbf{p}}_{\text{in}}$ and $\overline{\mathbf{p}}_{\text{out}}$ can be viewed as two probability distributions over all the possible class labels.

Step 2: Comparing KL-Divergence. The KL-Divergence [94] between two distributions P and Q is defined to be

$$D_{\text{KL}}(P \parallel Q) = \sum_i p_i \log \frac{p_i}{q_i}. \tag{4.9}$$

The adversary infers the membership of t by comparing $\mathbf{q}$'s KL-Divergence to $\overline{\mathbf{p}}_{\text{in}}$ and $\overline{\mathbf{p}}_{\text{out}}$, and accepts hypothesis H_1 if, and only if, $D_{\text{KL}}(\mathbf{q} \parallel \overline{\mathbf{p}}_{\text{out}}) > D_{\text{KL}}(\mathbf{q} \parallel \overline{\mathbf{p}}_{\text{in}})$.

In the experiment, we take $M = 5$. That is, for each target record t, we sample 5 datasets and train 10 shadow classifiers. 5 of the shadow classifiers are trained with t in the training set, and the other 5 are trained without t in the training set.

Frequency-Based Targeted Attacks. In a frequency-based targeted attack, the adversary trains the same shadow models as in the distance-based membership attack. However, instead of calculating the average of the predictions, the adversary counts the frequency that the predictions of classifiers satisfying hypothesis H_0 fall into the same bins as $c(\mathbf{x})$ as well as the frequency that the predictions of classifiers satisfying hypothesis H_1 fall into the same bins as $c(\mathbf{x})$. The adversary accepts the hypothesis under which predictions more often fall into the same bin as $c(\mathbf{x})$.

The first step of a frequency-based targeted attack is the same as the distance-based targeted attack. The adversary trains $2m$ shadow classifiers, m of which with t in training set. In the second step, for $1 \le i \le k$, the adversary calculates o_i^{in} as the number of shadow models that are trained with t in training set and gives the same predicted probability on class label y_i as the target dataset. Similarly, o_i^{out} is calculated as the number of shadow models that are trained without t in the training set and gives the same predicted probability on class label y_i as the target dataset.

Membership Attack	Accuracy	Precision	Recall	F1	Correlation	p-Value
Untargeted Attack	0.6680	0.6386	0.8500	0.7294	0.4864	< 0.01
Frequency-Based Attack	0.6257	0.5933	0.8253	0.7174	0.5052	< 0.01
Distance-Based Attack	0.8533	0.8470	0.9087	0.8768	0.7653	< 0.01

Table 4.1: Performance of Three Membership Attacks on NN-Purchase.

Finally the adversary estimates the following ratio:

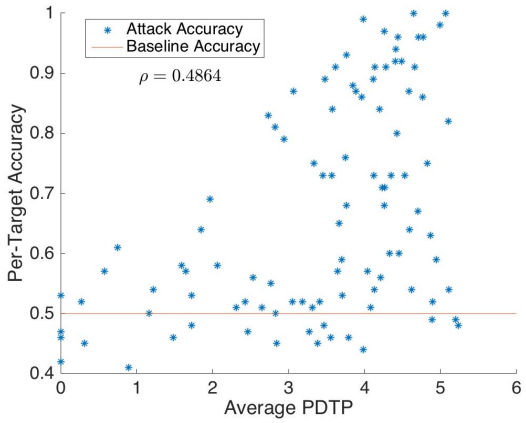
$$\frac{\Pr\left[t \in T \mid c(\mathbf{x}) = \mathbf{q}\right]}{\Pr\left[t \notin T \mid c(\mathbf{x}) = \mathbf{q}\right]} = \prod_{i=1}^{k} \frac{o_i^{\text{in}}}{o_i^{\text{out}}}. \tag{4.10}$$

The adversary accepts hypothesis H_1 iff $\frac{\Pr[t\in T|c(\mathbf{x})=\mathbf{q}]}{\Pr[t\notin T|c(\mathbf{x})=\mathbf{q}]} > 1$.

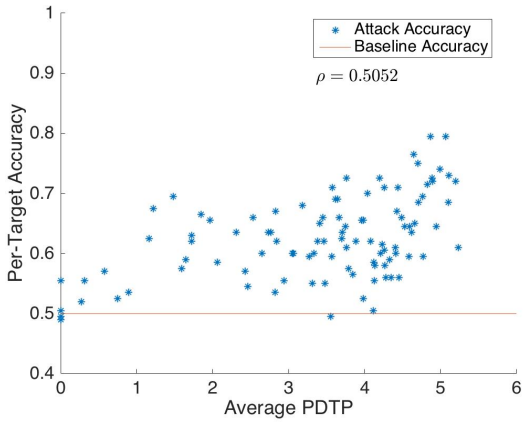
Like for the distance-based membership attack, we take $M = 5$ in the experiment and train 10 shadow models for each target record.

4.3.4 Evaluation Metrics

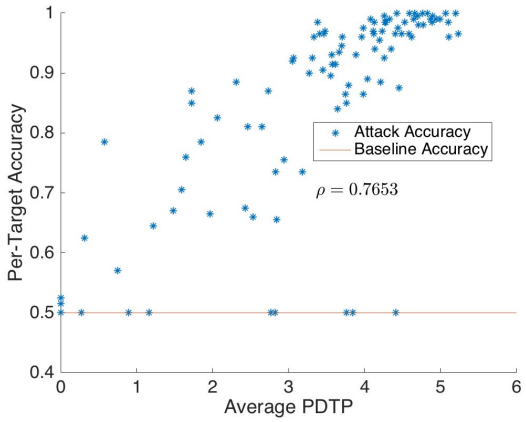
Multiple Iterations of Attacks. To evaluate their performance, we run 100 iterations of each membership attack. In each iteration, we partition the candidate set into two equal-sized parts: D_1 and D_2. First, we use D_1 as training set and D_2 as test set. A target classifier $\mathcal{A}(D_1)$ is trained on the training set and available for public queries. We randomly select 100 records out of D as the adversary's target records. For each target record t, we run a targeted attack with $\mathcal{A}(D_1)$ as target classifier. In each membership attack, the goal of the adversary is to predict whether $t \in D_1$ by querying $\mathcal{A}(D_1)$. Then, we switch the role of D_1 and D_2, and use D_2 as the training set and D_1 as test set. We then repeat the membership attack with $\mathcal{A}(D_2)$ as target classifier. This process ensures that the target record occurs once in the training set and once in the test set in each iteration so that the baseline accuracy is always 0.5. Since we can only calculate PDTP of a record t when t is in the training set, we measure the PDTP of t as $\text{PDTP}_{\mathcal{A},D_1}(t)$ if $t \in D_1$ and as $\text{PDTP}_{\mathcal{A},D_2}(t)$ if $t \in D_2$. Hence, in each iteration, we launch two membership attacks and get one PDTP measurement for each record in D based on definition 4.3. We repeat this process for 100 iterations. Ideally, one should calculate a record's average PDTP over 100 PDTP measurements. However, PDTP measurements over multiple datasets contain redundant information. As shown in figure 4.2, average PDTP taken over 10 measurements has approximately the same correlation with performance of membership inference attacks as average PDTP taken over 100 measurements. Therefore, to save time, we only take PDTP measurements in the first 10 iterations. For each target record, we use the average of its 10 PDTP measurements to estimate its overall



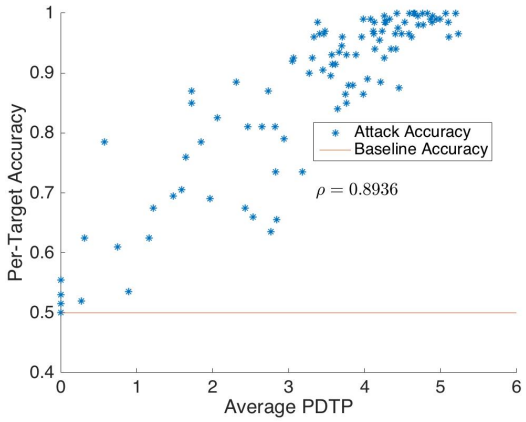
(a) Untargetted Membership Attack on NN-Purchase.



(b) Frequency-Based Membership Attack on NN-Purchase.



(c) Distance-Based Membership Attack on NN-Purchase.



(d) Maximum Per-Target Accuracy of Three Membership Attacks on NN-Purchase.

Figure 4.1: Membership Attacks on NN-Purchase.

membership risk.

Per-Target Attack Accuracy. We want to analyze the privacy risk of each record in D separately. That is, instead of looking at the membership attack accuracy on each training set, we are interested in the overall attack accuracy on a single record over the 200 membership attacks. Therefore, we propose *per-target attack accuracy* on a record t as the adversary's proportion of correct membership inference on t over all the attacks performed on t. For example, in the experiment, we launch 200 membership attacks on each record in D. Therefore, the per-target attack accuracy of a record is the number of correct membership inferences on that record divided by 200.

4.3.5 Results

Comparison of Different Attacks. First, we compare the performance of three membership attacks on neural networks trained on the purchase dataset. We train 200 neural network models over different training sets sampled from the same candidate set and use them as the target classifiers for membership attacks. All the target classifiers are overfitted to their training sets. The average training accuracy of all the target classifiers is 1, and the average test accuracy of all the target classifiers is 0.6434.

Figures 4.1a, 4.1b, and 4.1c show the per-target accuracy and average PDTP of each target record under three types membership attacks. Each point represents one target record in the candidate set. The horizontal axis is the average PDTP measurement of that target over 10 iterations of PDTP measurements. The vertical axis is the per-target accuracy of that target over 200 repetitions of membership attacks. A point's position on the horizontal axis shows its membership privacy risk estimated by PDTP. According to PDTP measurements, points on the right part of the figures have higher membership privacy risks compared to points on the left part of the figures. A point's position on the vertical axis shows its actual membership privacy risk under a given membership attack. Points on the top part of the figures are more vulnerable to the attack because the attack has higher accuracy on these records.

For each attack, we calculate the Pearson correlation coefficient between average PDTP and the per-target attack accuracy. We also calculate the p-value for testing the hypothesis of no correlation against the alternative hypothesis that there is a correlation between average PDTP and per-target attack accuracy. Table 4.1 shows the performance of each membership attack and their correlation coefficients with average PDTP. The performance of all three attacks has statistically significant correlations with the average PDTP. Figure 4.1 shows the accuracy of three types of membership attacks and their correlations with PDTP (ρ). We observe that attacks with higher accuracy also have higher correlation PDTP measurements. This correlation demonstrates PDTP's ability to identify potential membership privacy risks effectively.

Overall, distance-based targeted attacks have the highest accuracy. They outperform the untargeted attacks in the previous work [7] by approximately 19%. However, some records are more vulnerable to some types of membership attacks. For example, one record in the purchase dataset is immune to distance-based membership attacks which only achieve baseline accuracy, whereas the untargeted attack achieves accuracy of 0.94. This example demonstrates the insufficiency of estimating privacy risks based on one type of attack. Even a strong attack may fail to identify some of the vulnerabilities that can be used by other

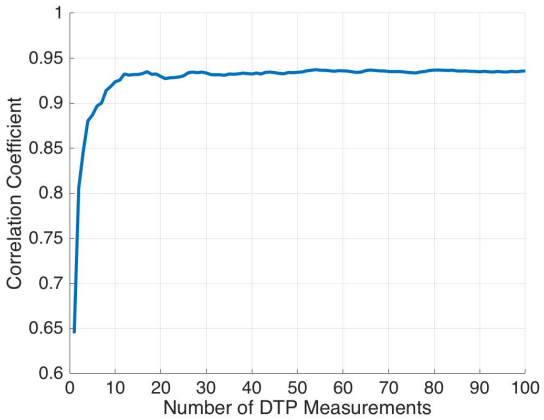


Figure 4.2: Correlation between Average PDTP and Membership Attack Accuracy.

Model	Accuracy	Precision	Recall	F1 Score	PDTP	Corr.	p-value
NN-Purchase	0.8533	0.8470	0.9087	0.8768	3.4019	0.7653	< 0.01
NB-Purchase	0.5958	0.6945	0.4038	0.5107	0.9027	0.9239	< 0.01
LR-Purchase	0.7888	0.7314	0.9187	0.8144	2.8917	0.8138	< 0.01
NN-Adult	0.5340	0.5311	0.4402	0.4356	0.5847	0.4588	< 0.01
NB-Adult	0.5128	0.5876	0.1027	0.1748	0.0299	0.5166	< 0.01
LR-Adult	0.5134	0.5130	0.3818	0.4378	0.1460	-0.0008	0.9343

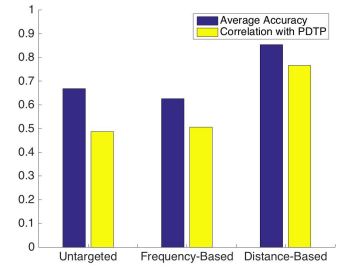
Table 4.2: Performance of Distance-Based attack on Different Target Models.

attacks. Figure 4.1d shows the maximum per-target accuracy among three membership attacks. The Pearson correlation coefficient between the maximum accuracy and PDTP measurements is 0.8936. This is very strong correlation. Among all records with PDTP greater than 1, 84.62% of them have maximum per-target accuracy higher than 0.8, and all of them have maximum per-target accuracy higher than 0.6. This result supports the DTP-1 hypothesis that classifiers with DTP above 1 should not be published.

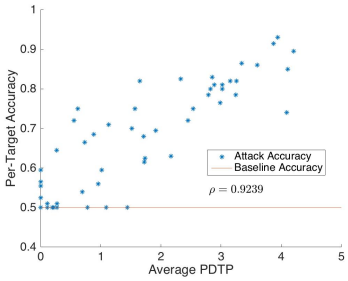
Privacy Risks of Different Models and Datasets. To compare the privacy risks of different datasets and classifiers, we use PDTP measurements on NN, NB, and LR classifiers learned on the adult and purchase datasets. We use distance-based membership attack because it has higher overall accuracy. Table 4.2 shows the performance of each target model, the per-target accuracy of membership attacks, and its correlation with PDTP measurements.

The results of membership attacks on NB and LR models trained on the purchase dataset are shown in Figure 4.3b and Figure 4.3c. Both of the two classifiers have records with PDTP higher than 1, and most of these records are vulnerable to distance-based membership attacks. The accuracy of the attacks is highly correlated with PDTP measurements.

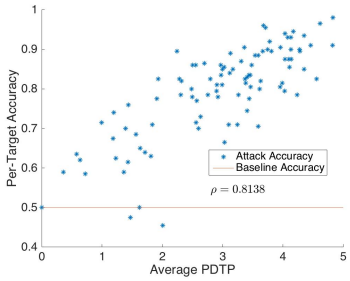
The results of membership attacks on NN, NB, and LR models trained on the adult dataset



(a) Comparison between Three Membership Attacks on NN-Purchase10



(b) Distance-Based Membership Attack on NB-Purchase



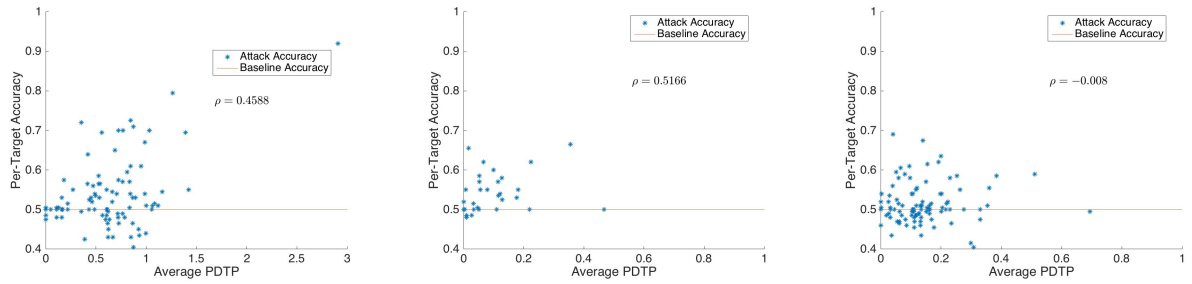
(c) Distance-Based Membership Attack on LR-Purchase

Figure 4.3: Membership Attacks on classifiers learned on purchase dataset

are shown in Figure 4.4a, Figure 4.4b, and Figure 4.4c. Unlike the purchase dataset, the adult dataset has fewer classes and features which help improve the generalizability of models learned on this dataset. The training and test accuracy reflects good generalizability of all three models learned on the adult dataset. The distance-based membership attacks also have worse performance on the adult dataset, indicating better membership privacy. However, even if the average PDTP measurement is relatively low for all three models, the PDTP for some records is greater than 1 with the neural network model learned on the adult dataset indicating high membership privacy risk. This risk is also reflected by the high per-target accuracy of distance-based membership attack on some of the records with high PDTP. Therefore, good generalizability is not always sufficient for protecting membership privacy. It is possible that a model is not overfitted on the training set, but still captures some private information of some records in the training set. However, this privacy risk can be discovered by measuring PDTP for each record in the training set.

For NB and LR models trained on the adult dataset, we did not find any records with PDTP greater than 1, and the per-target attack accuracy of the distance-based attack is smaller than 70% for all target records. This result shows that state-of-art membership inference attacks do not work well on these models, and the PDTP measurements do not indicate high privacy risk for any of the records. The correlations between PDTP measurements and per-target attack accuracy is lower compared to attacks with better performance.

Multiple PDTP Measurements. In the previous experiments, for each target t, we use the average of 10 PDTP measurements on t to estimate the PDTP of t over all the target classifiers. To study how the number of PDTP measurements influence our estimation of the membership privacy risk of t, we take the Naive Bayes classifier trained on the purchase dataset as the target model perform a distance-based membership attack. We gradually increase the



(a) Distance-Based Membership Attack on NN-Adult. (b) Distance-Based Membership Attack on NB-Adult (c) Distance-Based Membership Attack on LR-Adult.

Figure 4.4: Membership Attacks on classifiers learned on adult dataset.

number of PDTP measurements from 1 to 100 and calculate the average PDTP's correlation with the accuracy of membership attacks. Figure 4.2 shows that the correlation between average PDTP and accuracy of membership attacks increases as we increase the number of PDTP measurements. The correlation coefficient stabilizes after around 10 measurements.

4.4 PROTECTIONS AGAINST INDIRECT MEMBERSHIP ATTACKS

In this section, we investigate the risk of indirect membership attacks where the adversary queries the classifier for features other than the target record.

4.4.1 Risk

In the previous experiments, we assume that the best way of doing a membership attack is to launch a direct attack by querying the target record. However, is it possible that, for some classifier c, there exists a query $\mathbf{x} \neq \mathbf{x}^{(t)}$ so that $c(\mathbf{x})$ leaks more private information than $c(\mathbf{x}^{(t)})$? That is, can an indirect membership attack outperform any direct membership attacks? Although it is hard to design a good indirect membership attack for classifiers discussed in Section 4.3, this risk of indirect membership attacks can be demonstrated with a carefully designed classifier that encodes membership information of one specific record.

Let $c = \mathcal{A}(T)$ be a classifier learned on a training set T with machine learning algorithm $\mathcal{A}$. Instead of releasing c, we construct a classifier c^* as follows:

$$c^*(\mathbf{x}) = \begin{cases} c(\mathbf{x}) & \text{if} \quad \mathbf{x} \neq \mathbf{0} \\ \mathbf{1} & \text{if} \quad \mathbf{x} = \mathbf{0} \quad \text{and} \quad t \in T \\ \mathbf{0} & \text{if} \quad \mathbf{x} = \mathbf{0} \quad \text{and} \quad t \notin T. \end{cases} \tag{4.11}$$

Assume $\mathbf{x}^{(t)} \neq \mathbf{0}$, apparently, an indirect membership attack with query $\mathbf{x} = \mathbf{0}$ gives more information about the target t compared to a direct membership attack with query $\mathbf{x}^{(t)}$. This example shows that for some classifiers, indirect attacks can outperform direct attacks for some records. Therefore, a record can have high membership privacy risk even if its PDTP measurement is low.

Clearly, c^* is not representative of a real-life machine learning model, especially when the model is trained by the data owner who wants to protect against privacy leakage. However, to achieve a stronger privacy guarantee, we need to study the potential risk of indirect membership attacks and prevent models from leaking "side channel" information about records in their training sets.

4.4.2 Training Stability

To protect against indirect membership attacks, we need a way of calculating $\text{DTP}_{\mathcal{A},T}(t)$ without the need of brute forcing the whole feature space X^m. Since we can already efficiently calculate $\text{PDTP}_{\mathcal{A},T}(t)$, a natural approach is to bound $\text{DTP}_{\mathcal{A},T}(t)$ based on $\text{PDTP}_{\mathcal{A},T}(t)$. We call this property training stability.

Definition 4.5. (Training Stability) A classification algorithm $\mathcal{A}$ is *δ-training stable* on dataset T if there exists a constant $\delta > 1$, so that for all $t \in T$ with $p_{A(T\setminus\{t\})}(y^{(t)} \mid \mathbf{x}^{(t)}) > 0$ and $p_{A(T)}(y^{(t)} \mid \mathbf{x}^{(t)}) > 0$, for all $\mathbf{x} \in X^m$, for all $y \in Y$, let

$$\gamma_t = \max(\delta, \frac{p_{\mathcal{A}(T)}(y^{(t)} \mid \mathbf{x}^{(t)})}{p_{A(T\setminus\{t\})}(y^{(t)} \mid \mathbf{x}^{(t)})}, \frac{p_{\mathcal{A}(T\setminus\{t\})}(y^{(t)} \mid \mathbf{x}^{(t)})}{p_{A(T)}(y^{(t)} \mid \mathbf{x}^{(t)})}), \tag{4.12}$$

we have

$$p_{\mathcal{A}(T)}(y \mid \mathbf{x}) \leq \gamma_t p_{A(D\setminus\{t\})}(y \mid \mathbf{x}), \tag{4.13}$$

and

$$p_{\mathcal{A}(T)}(y \mid \mathbf{x}) \geq \gamma_t^{-1} p_{A(D\setminus\{t\})}(y \mid \mathbf{x}). \tag{4.14}$$

Given an algorithm that is δ-training stable on T, for all $t \in T$, the ratio between the predictions of two classifiers $\mathcal{A}(T)$ and $\mathcal{A}(T \setminus \{t\})$ is bounded either by the ratio between their predictions on the query $(\mathbf{x}^{(t)}, y^{(t)})$ or by a parameter δ.

If an algorithm $\mathcal{A}$ is δ-training stable on T, $\text{DTP}_{\mathcal{A},T}(t)$ can be calculated by measuring $\text{PDTP}_{\mathcal{A},T}(t)$, which is much more efficient.

Theorem 4.1. *If a record $t \in T$ is ϵ-PDTP with classification algorithm $\mathcal{A}$ and dataset T, and $\mathcal{A}$ is δ-training stable on T, we have t is ϵ'-DTP with $\mathcal{A}$ and T, where $\epsilon' = \max(\epsilon, \ln \delta)$.*

On the one hand, δ-training stability is a desirable property from a privacy perspective because it reduces the computational cost of estimating the influence of an individual record on the learned classifier. On the other hand, δ-training stability is also a metaphor of learning in real life. For example, If a professor explains an example question in class, he expects the students to do well on similar questions in the exam. If the exam contains a question that is the same as the example question explained in class, most students are expected to answer it correctly. Similarly, suppose we have a classifier $\mathcal{A}(T \setminus \{t\})$ and an additional record t. By adding t into the training dataset, we expect the classifier to have better performance on t or records similar to t. This can be viewed as a metaphor of learning in real life.

4.4.3 Training Stability of Classifiers

With the aforementioned intuitions in mind, we study the training stability of some commonly used classifiers. However, due to the complexity and variability of different machine learning algorithms, we cannot cover all well-known classifiers in this section. Table 4.3 shows the training stability of some commonly used classifiers.

Bayes Inference Classifiers. For a Bayes inference classifier $\mathcal{A}(T)$, the prediction $p_{\mathcal{A}(T)}(y \mid \mathbf{x})$ is given by the conditional probability of class label y given feature vector $\mathbf{x}$ in the training dataset T.

Proposition 4.2. *Bayes inference algorithm is δ-training stable for $\delta = \frac{4}{3}$ on any training dataset.*

Naive Bayes Classifiers. Naive Bayes classifiers make predictions using Bayes theorem and assume conditional independence [91].

Proposition 4.3. *Let T be a training dataset with m features and n examples. Let $y_{\min}$ be the least supported class label in T. Let $n_{y_{\min}}$ be the number of examples with class label $y_{\min}$. Naive Bayes algorithm is δ-training stable for*

$$\delta = \left(\frac{n_{y_{\min}}}{n_{y_{\min}} - 1}\right)^{m-1} \frac{n}{n-1}. \tag{4.15}$$

If T is a large training dataset, there would be a large number of training examples with class label $y_{\min}$. Therefore, δ would be close to 1 for a large dataset T, and the maximum ratio between predictions of $\mathcal{A}(T)$ and $\mathcal{A}(T \setminus \{t\})$ is determined by the ratio between their predictions on the query $(\mathbf{x}^{(t)}, y^{(t)})$.

Naive Bayes classification is often used with Laplace smoothing [95]. When conditional probabilities are estimated from the training dataset, a small constant is added to both the numerator and denominator to get a "smoothed" version of the prediction. The constant is determined by the number of possible values for each attribute. Suppose each attribute in $\mathbf{x}$ has at most v possible values. When calculating the conditional probability of an attribute given the class label, the numerator is increased by 1, and the denominator is increased by v. Therefore, naive Bayes classification algorithm with Laplace smoothing is δ-training stable with a slightly different δ compared to the original naive Bayes classification.

Proposition 4.4. *Let T be a training dataset with m features and n examples. Suppose each element in the feature vector has at most v possible values. Let $y_{\min}$ be the least supported class label in T. Let $n_{y_{\min}}$ be the number of examples with class label $y_{\min}$. Naive Bayes with Laplace smoothing is δ-training stable on T for*

$$\delta = \left(\frac{n_{y_{\min}} + v}{n_{y_{\min}}}\right)^{m-1} \frac{n}{n-1}. \tag{4.16}$$

Linear Statistical Queries Classifiers. Linear statistical queries (LSQ) classifiers are proposed as a generalization framework for naive Bayes, Bayesian network, and Markov models [96].

Let $\chi : X^m \to \{0, 1\}$ be a feature function that maps a feature vector into a binary value. This representation is useful for features depending on more than one element in $\mathbf{x}$ (for example, $\chi(\mathbf{x}) = 1$ iff $x_1 = 1$ and $x_2 = 1$). A *statistical query* $\hat{P}^T_{[\chi,y]}$ gives the probability of all the examples with feature $\chi(\mathbf{x}) = 1$ and class label y in the training dataset.

A linear statistical queries (LSQ) classifier is a linear discriminator over the feature space, with coefficients calculated by statistical queries. For the convenience of discussion, we review the following definition of LSQ classifier for binary classification:

Definition 4.6 (Linear Statistical Queries classifier [96])**.** Let $\mathcal{X}$ be a class of features. Let $f_{[\chi,y]}$ be a function that depends only on the values $\hat{P}^T_{[\chi,y]}$ for $\chi \in \mathcal{X}$. A *linear statistical queries (LSQ) hypothesis* predicts $y \in \{0, 1\}$ given $\mathbf{x} \in X^m$ when

$$y = \arg\max_{y \in \{0,1\}} \sum_{\chi \in \mathcal{X}} f_{[\chi,y]}(\hat{P}^T_{[\chi,y]})\chi(\mathbf{x}). \tag{4.17}$$

We define a family of *log coefficient functions* $\mathcal{F}_{\log}$ that contains all the functions $f_{[\chi,y]}$ that calculate the log of a probability or conditional probability in the training dataset. For example, suppose $\mathcal{A}$ is a naive Bayes classification algorithm with m feature attributes. $\mathcal{A}$

can be written as an LSQ classification algorithm with m features: $\chi_0 \equiv 1$, and $\chi_j = x_j$ for $1 \leq j \leq m$, where

$$f_{[\chi_0,y]}(\hat{P}^T_{[\chi_0,y]}) = \log \hat{P}^T_{[1,y]}, \tag{4.18}$$

and for $1 \leq j \leq m$,

$$f_{[\chi_j,y]}(\hat{P}^T_{[\chi_j,y]}) = \log {\hat{P}^T_{[x_j,y]}}/{\hat{P}^T_{[1,y]}}. \tag{4.19}$$

$f_{[\chi_0,y]}$ is a log function of the prior probability of y, and $f_{[\chi_j,y]}$ is a log function of the conditional probability of x_j given y. Therefore, the coefficient functions for naive Bayes belong to the family of log coefficient functions. Similarly, log coefficient functions are also used in Bayes network and Markov model.

When $f_{[\chi,y]} \in \mathcal{F}_{\log}$, the sum of $f_{[\chi,y]}$ is equivalent to the product of the corresponding probabilities. Therefore, in addition to returning the most likely label, an LSQ classifier $\mathcal{A}(T)$ is also a probabilistic classifier that returns the following predicted probability:

$$\begin{aligned} p_{\mathcal{A}(T)}(y \mid \mathbf{x}) &= e^{\sum\limits_{\chi \in \mathcal{X}} f_{[\chi,y]}(\{\hat{P}^T_{[\chi,y]}\})\chi(\mathbf{x})} \\ &= \prod_{\chi \in \mathcal{X}} e^{f_{[\chi,y]}(\{\hat{P}^T_{[\chi,y]}\})\chi(\mathbf{x})}. \end{aligned} \tag{4.20}$$

Each term $e^{f_{[\chi,l]}(\{\hat{P}^T_{[\chi,l]}\})\chi(\mathbf{x})}$ in Equation (4.20) is equivalent to calculating a probability or a conditional probability in the training dataset T using Bayes inference, therefore is $\frac{4}{3}$-training stable according to proposition 4.2. Consequently, we have the following proposition for LSQ probabilistic classification algorithms:

Proposition 4.5. *If $\mathcal{A}$ is an LSQ probabilistic classification algorithm with $f_{[\chi,y]} \in \mathcal{F}_{\log}$ for all $\chi \in \mathcal{X}, y \in Y$, $\mathcal{A}$ is δ-training stable on any training dataset with $\delta = (\frac{4}{3})^{|\mathcal{X}|}$.*

For naive Bayes classification algorithm, since each attribute is an independent feature, with $M-1$ feature attributes, $|\mathcal{X}|$ equals M . Compared to proposition 4.3, proposition 4.5 provides a looser but more generalized bound on δ-training stability for naive Bayes classification algorithm. This bound does not depend on the records in the training dataset T. For Bayesian network and Markov models, $|\mathcal{X}|$ equals to the layer of dependencies in the network. $|\mathcal{X}|$ gets larger when the network structure gets more complicated.

Decision Trees. Some decision trees, such as ID3 and C4.5, construct the structure of the tree by calculating information gain of each potential partition of attributes[97]. This approach makes achieving δ-training stability difficult because when an example is removed from the training dataset, it is hard to predict its influence on the structure of the tree. For example, removing one example may change the splitting point with the highest information

Training Stable Classifiers	Training Stability Unknown	Non-Training Stable Classifiers
Bayes Inference Classifiers,	Support Vector Machines	k-Nearest Neighbors
Linear Statistical Queries	Neural Networks	
Naive Bayes Classifiers	Logistic Regressions	
Random Decision Trees	...	

Table 4.3: Training Stability of Different Classifiers.

gain, so that the structure of $\mathcal{A}(T)$ and $\mathcal{A}(T \setminus \{t\})$ are completely different.

However, when the structure of a decision tree is independent of its training dataset, its prediction is equivalent to the conditional probability of y given a subset of attributes determined by the leaves of the tree. Therefore, a single decision tree with structure independent of its training dataset is $\frac{4}{3}$-training stable.

A random decision tree classifier is a classifier constructed by aggregating K randomly generated decision trees with structures independent of the training dataset[98]. Random decision trees have better privacy properties because the structure of the trees do not leak private information about the training set. Previous work has shown that a large amount of noise is needed to make ID3 differentially private while it is more practical to achieve differential privacy for a random decision tree [99].

If the predictions of the random decision tree classifier is aggregated in a way that preserves the training stability, the random decision tree classifier is also training stable.

Proposition 4.6. *Let $\mathcal{A}_K$ be a random decision tree classification algorithm with K randomly generated decision trees. Given a query $(\mathbf{x}, y)$, let $p_1(y \mid \mathbf{x}), p_2(y \mid \mathbf{x}), \ldots, p_K(y \mid \mathbf{x})$ be the predictions given by each random decision tree. $\mathcal{A}_K$ is $\left(\frac{4}{3}\right)$-training stable, if it computes the prediction as follows:*

$$p_{\mathcal{A}(T)}(y \mid \mathbf{x}) = e^{\frac{1}{K}\sum_{j=1}^{K} \log(p_j(y|\mathbf{x}))}. \tag{4.21}$$

***k*-Nearest Neighbors.** k-nearest neighbors (k-NN) classification [100] is an instance-based learning algorithm. Instead of constructing a model from the training dataset, all examples in the training dataset are saved and all computations are deferred until classification. When responding to a query $(\mathbf{x}, y)$, predictions are made by approximating locally from a few examples close to the query. Unlike the aforementioned classifiers, k-NN is not training stable for any δ.

For simplification, suppose $\mathcal{A}$ is a 1-nearest neighbor classification algorithm. Let $(\mathbf{x}^{(t)}, y^{(t)})$, $(\mathbf{x}_1, y_1)$ and $(\mathbf{x}_2, y_2)$ be three examples in a training dataset. $(\mathbf{x}_1, y_1)$ is the nearest neighbor of $(\mathbf{x}^{(t)}, y^{(t)})$ when t itself is not in the training dataset. Let $(\mathbf{x}', y')$ be a point whose nearest neighbor in the training dataset is $(\mathbf{x}^{(t)}, y^{(t)})$ and second nearest neighbor is $(\mathbf{x}_2, y_2)$. Suppose $y^{(t)} = y_1 = y' \neq y_2$. When $t \in T$, the classifier $\mathcal{A}(T)$ will predict the class label as $y^{(t)}$ for both

features $\mathbf{x}^{(t)}$ and $\mathbf{x}'$. When t is removed from the training dataset, the classifier $\mathcal{A}(T \setminus \{t\})$ will still predict the class label as $y^{(t)}$ for feature $\mathbf{x}^{(t)}$ because of point $(\mathbf{x}_1, y_1)$. However, $\mathcal{A}(T \setminus \{t\})$ will predict the class label for $\mathbf{x}'$ as y_2 since it is closest to $(\mathbf{x}_2, y_2)$ when $(\mathbf{x}^{(t)}, y^{(t)})$ is removed. Consequently, when t is removed from the training dataset, the prediction for t remains unchanged, but the prediction for a neighboring point $(\mathbf{x}', y')$ is greatly influenced. If we calculate the probability given by the classifier, we have

$$p_{\mathcal{A}(T)}(y' \mid \mathbf{x}') = 1 \quad \text{and} \quad p_{\mathcal{A}(T\setminus\{t\})}(y' \mid \mathbf{x}') = 0, \tag{4.22}$$

while

$$\frac{p_{\mathcal{A}(T)}(y^{(t)} \mid \mathbf{x}^{(t)})}{p_{\mathcal{A}(T\setminus\{t\})}(y^{(t)} \mid \mathbf{x}^{(t)})} = 1. \tag{4.23}$$

As k increases, the probability of the aforementioned case drops. However, there is always a possibility that, when an exampled t is removed from the training dataset T, for some queries $(\mathbf{x}, y)$, the prediction $p_{\mathcal{A}(T)}(y \mid \mathbf{x})$ drops from 1 to 0, while the prediction $p_{\mathcal{A}(T)}(y^{(t)} \mid \mathbf{x}^{(t)})$ remains unchanged. Therefore, k-nearest neighbors classification algorithm is not training stable.s

4.4.4 An Upper Bound on DTP

For non-training stable classifiers or classifiers with unknown training stability, the DTP metric cannot be directly calculated based on PDTP measurements. However, it is still possible to estimate an upper bound for DTP based on Lipschitz conditions.

Given a classification algorithm $\mathcal{A}$, the set of possible classifiers learned by $\mathcal{A}$ can be abstracted as a class of functions $\{C_u, u \in \mathcal{U}\}$, where $u \in \mathcal{U}$ is a d-dimensional vector that specifies the trainable parameters in the classifier and $\mathcal{U} \subseteq \mathbb{R}^d$. Without loss of generality, we assume that C_u maps a feature vector $\mathbf{x}$ to a vector of predicted log probabilities of each class labels $y \in Y$. That is, $C_u(\mathbf{x}) = \big(\log p_{\mathcal{A}(T)}(y_1 \mid \mathbf{x}), \log p_{\mathcal{A}(T)}(y_2 \mid \mathbf{x}), \ldots, \log p_{\mathcal{A}(T)}(y_k \mid \mathbf{x})\big)$.

We assume that for all $\mathbf{x} \in X^m$, $C_u(\mathbf{x})$ is L-Lipschitz bounded under infinity norm with respect to u. That is, $|C_u(\mathbf{x}) - C_{u'}(\mathbf{x})|_\infty \leq L\,|u - u'|_\infty$. Based on Lipschitz condition, to calculate $\mathrm{DTP}(\mathcal{A}, T)$, it is enough to measure the change of model parameters when one training example is removed.

Let u_T be the model parameters learned on dataset T and $u_{T\setminus\{t\}}$ be the parameters learned on $T \setminus \{t\}$. The following theorem gives an upper bound of $\mathrm{DTP}(\mathcal{A}, T)$:

Theorem 4.7. *If $\mathcal{A}(T)$ is an L-Lipschitz bounded classifier, then $DTP_{\mathcal{A},T}(t)$ is upper bounded by $L \cdot \max_{t\in T} \left|u_T - u_{T\setminus\{t\}}\right|_\infty$.*

4.5 REDUCING DTP

According to the DTP-1 hypothesis, it is unsafe to release a classifier if any record in its training set has DTP greater than 1. However, what should the data owner do if only a small number of records in the training dataset violate this hypothesis? It is unsafe to release the classifier since it contains records vulnerable to membership inference attacks using techniques like those in Section 4.3. But it is natural to ask: *Can removing high-risk records from the training set mitigate the membership privacy risk?* The interesting answer is sometimes yes and sometimes no!

Let us consider a specific example of how removing high-risk records can influence DTP. The examples in Section 4.3 are not ideal because the classifiers trained on the adult dataset already have low privacy risks, while the classifiers trained on the purchased dataset are so risky that they are unlikely to be mitigated by simple mechanisms. We therefore consider a fresh example that fails to satisfy DTP-1, but not by much. To do this we train a naive Bayes classifier on the 2013 American Community Survey (ACS) dataset. This dataset has similar attributes as the adult dataset since the adult dataset was sampled and cleaned from the 1994 Census dataset. We restrict to four attributes: Age (AGEP), Marital Status (MAR), Race (RAC1P), and Gender (SEX). As we did with the adult dataset, we use the salary class (> 50K or ≤ 50K) as the class attribute. We use all 1.6 million records as our training set. The DTP of the full dataset is 3.09, indicating vulnerability to membership inference attacks.

To reduce the DTP in the dataset, we perform the following simple experiment: First, we measure DTP of each record in the training set and sort all the records in the decreasing order of their DTP measurements. Intuitively, the records are sorted in the decreasing order of their (initial) privacy risks. Next, we remove these high-risk records from the training set one at a time. After each record is removed, we re-calculate the DTP of all records remaining in the training set and estimate the resulting privacy risk as the highest DTP in this reduced training set. Figure 4.5 shows the change of maximum DTP in the training set when these high-risk records are removed. Removing the record with the highest risk reduces the highest DTP in training set from 3.09 to 0.65, greatly reducing the classifier's vulnerability to membership attacks and achieving DTP-1. However, one must not get greedy and think that removing the next individual will reduce the risk even further. Doing this takes the DTP back to around 3. Why? Because, unlike the first individual removed, this second record apparently is needed to decrease another record's influence on the target classifier. Indeed, removing further individuals appears to lead to collections of individuals that rely on each other to keep DTP down. Their successive removal creates the sawtooth pattern seen in Figure 4.5. Based on this observation, we recommend removing high-risk examples as a way

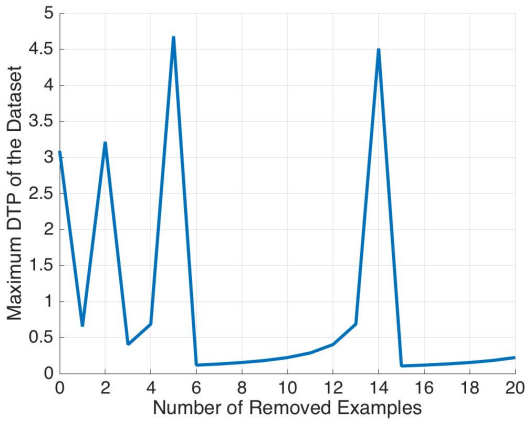


Figure 4.5: Effects of Removing High-Risk Records.

of reducing DTP and mitigating against membership attacks when only a few examples in the training set have high privacy risks. Better understanding of how to reduce DTP is a promising target for future research.

4.6 DISCUSSION

In this section, we discuss a few interesting points related to DTP.

Difference between DTP and DP. When feasible, using differential privacy during training is a good strategy to mitigate the risk of publishing the model. However, there are cases when differential privacy cannot be used; either because there is no appropriate training mechanism or because the data owner cannot afford to add noise to their models (e.g., in the medical domain). Therefore, we need a strategy to estimate the privacy risk of the model when no privacy protections are added. Note that even when differential privacy is used, DTP can still be used to estimate the privacy risks before applying differential privacy. With the DTP measurements, the data owner can understand how much he benefits from using differential private mechanisms. This information helps balance the trade-off between utility and privacy.

Unlike DP, DTP is a privacy metric instead of a privacy protection mechanism. When a machine learning model does not satisfy differential privacy for any ϵ, little is known about its privacy risk. However, the metric $\text{DTP}_{\mathcal{A},T}(t)$ outputs a value of ϵ for *any* target record t and any *any* classifier $\mathcal{A}(T)$.

Difference between DTP and Membership Attacks. In Section 4.3, we show that DTP measurements correlate with the accuracy of different membership attacks. However,

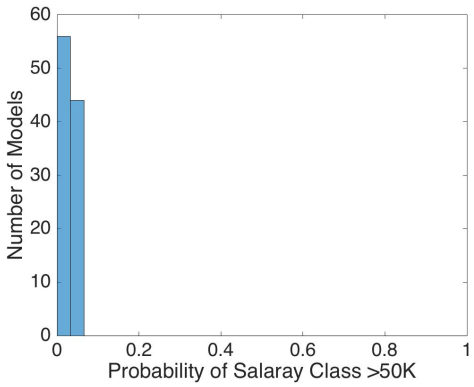
the measurement of DTP *cannot* be replaced by running a series of membership attacks. First, it is computationally inefficient to simulate all possible membership attacks. Moreover, no matter how much computational power a data owner has, there may always exist an adversary with superior computational capability. Second, we cannot rule out the possibility that the adversary knows a stronger a membership attack than the data owner. As demonstrated in section 4.3, a record immune to the distance-based membership attack can be vulnerable to another attack—even a weaker one, overall. Therefore, using a general privacy metric like DTP to estimate membership privacy is preferable.

Privacy Risks of Non-Training Stable Classifiers. In Section 4.4, we prove that naive Bayes, random decision trees, and linear statistical queries satisfy training stability but k-NN provably does not. However, we do not know whether classifiers such as neural networks and SVMs are training stable. We leave the task of investigating this question for future work.

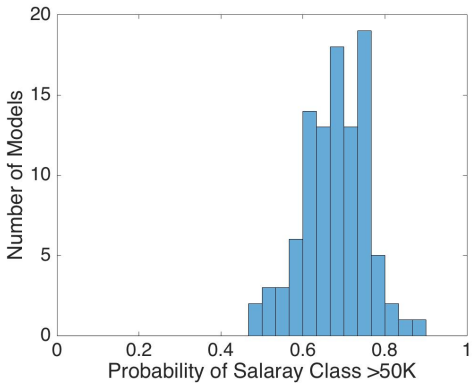
Remark that although measuring DTP is computationally infeasible for non-training stable models, this does not mean that DTP metrics are useless for these models. Indeed, as shown in Section 4.3 PDTP measurements have high correlations with direct membership attacks. The drawback for non-training stable algorithms is their potential vulnerability to indirect membership attacks. As future work, we plan to study indirect membership attacks and ways to mitigate them.

Although DTP doesn't provide a theoretical privacy guarantee like DP, we find that it is highly correlated with the performance of state-of-the-art membership inference techniques. Unlike DP, which bounds the change in the probability of observing an output when a record is removed, DTP bounds the magnitude of the difference caused by removal of a record. In practice, if the magnitude of this difference is small, it is indistinguishable from the difference caused by other uncertain factors from the adversary's perspective. When attacking machine learning models, these are at least two sources of uncertainty.

First, in models like neural networks, some parameters such as weights are initialized randomly. Different initialization states may cause the model to be converged to different local optimals, so models trained on the same dataset can give slightly different predictions on the same record. If a record's DTP is small enough to be indistinguishable from the difference caused by random initialization, the record has little privacy risk. Figure 4.6 shows the variation in model prediction caused by random initialization. In the experiment, we train 100 neural network models on the same training set with 10000 records uniformly sampled from the UCI Adult dataset. We calculate each model's prediction on two individuals and plot the histogram of the predicted probability that the individual has annual salary greater

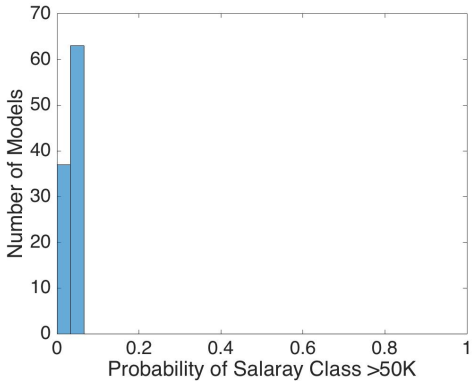


(a) Prediction on an individual in low salary class

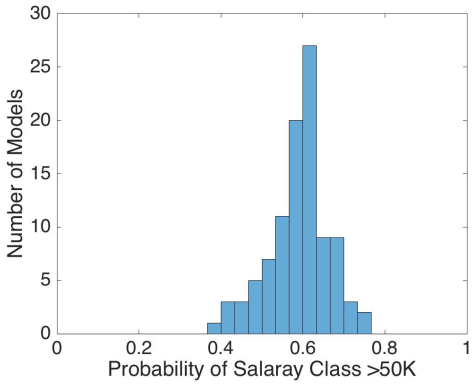


(b) Prediction on an individual in high salary class

Figure 4.6: Prediction variation caused by random initialization



(a) Prediction on an individual in low salary class



(b) Prediction on an individual in high salary class

Figure 4.7: Prediction variation caused by random sampling

than 50K.

Second, besides the target record, the adversary is also uncertain about what other records are included in the training dataset. The occurrence of unexpected training records can introduce small variations in the model's predcitions. If a record's DTP is small enough to be indistinguishable from the variation caused by random sampling, the target has little privacy risk. Figure 4.7 shows the variation in model prediction caused by random sampling. In the experiment, we train 100 classifiers on 100 different training datasets uniformly sampled from the same population. We calculate each model's prediction on two individuals and plot the histogram of the predicted probability that the individual has annual salary greater than 50K.

4.7 OPEN QUESTIONS

Experimental results suggest that DTP is a good predictor of the performance of state-of-art membership inference attacks. However, it remains an open question if records with low DTP are always safe from membership inference attacks. In this section, we discuss two potential privacy risks for low-DTP records.

DTP-1 Hypothesis. In this chapter, we use the DTP-1 hypothesis as guidance to identify records and classifiers with high privacy risk. By experimenting with state-of-art membership inference attacks on machine learning models, we find that when a training record only has a very small influence on the prediction of a classifier, this small influence is likely to be indistinguishable from the variation in prediction due to random sampling of the training records or random initialization of the weight vectors before training. We use DTP-1 hypothesis as a rule-of-thumb for determining whether the influence of a training record is smaller than the influence of other factors unknown to the adversary, such as randomization in the training algorithm and existence of unexpected records in the training set. However, in practice, even when DTP is smaller than 1, the influence of these uncertain factors can be smaller than the influence of the training record. Therefore, satisfying the DTP-1 hypothesis cannot guarantee that records with DTP smaller than 1 have no privacy risks. With a better understanding on the adversary's background knowledge and the influence of randomness in machine learning algorithms, it may be possible to determine a finer threshold for a safe DTP.

Risk of Indirect Attacks and Multiple Queries. PDTP measures the privacy risk of directly querying the target record. Based on experimental validations, we find that training records with low PDTP are less likely to be vulnerable to direct attacks. However, models that are not training stable have a potential of leaking the record's membership information through other queries, and an adversary may use this information to perform an indirect attack. Although we do not know of any practical indirect attacks, it remains an open question to analyze the training stability of some machine learning models and to design indirect attacks for models that are not training stable.

Another challenge is to analyze the risk of allowing an adversary to get predictions of multiple queries from the same machine leanrning model. DTP measures the privacy risk for a single query. However, if an adversary is allowed to query the target model multiple times, he may accumulate more information about the target record t. We leave for future work the study of how this accumulation of information can be used to design stronger membership inference attacks. Specifically, there are two open questions: (1) *How do we select multiple queries whose results indicate the membership of a target record?* (2) *How do we estimate an*

upper bound on the accuracy of membership inference when an adversary can submit unlimited number of queries to the model?

4.8 CONCLUSIONS

In this work, we propose differential training privacy (DTP) as an empirical metric to estimate the privacy risk of publishing a classifier. DTP estimates the privacy risk of a training record by measuring its influence on the predictions of machine learning models. A large DTP indicates that the record's influence is strong enough to indicate its presence in the training dataset. We measure DTP of popular machine learning models including neural networks, Naive Bayes, and logistic regressions. We compare these measurements with the accuracy of different types of membership inference attacks, including the most effective one in prior works. Experimental results demontrate that DTP is both efficient and effective in estimating privacy risks. Specifically, our attacks have at most 66.5% accuracy (baseline: 50%) on classifiers with DTP-values under 0.5 and almost always over 90% accuracy on classifiers with DTP larger than 4. Based on these results, we propose DTP-1 hypothesis as a rule-of-thumb criterion for publishing a classifier: *if a classifier has a DTP value above* 1*, it should not be published.*

Although DTP has a high correlation with the accuracy of a membership attack, it provides no guarantee about a record's privacy protection. Specifically, we propose two potential privacy leakages for records with low DTP. First, a low-DTP record is vulnerable to membership inferences when the model's predictions on it are unlikely to be influenced by other records or random initializations. Second, the membership of a low-DTP record might be leaked by indirect queries or the combination of multiple queries. This observation can serve as a new direction for designing stronger membership inference attacks and defenses.

CHAPTER 5: BLACK-BOX PROPERTY INFERENCE ATTACKS ON MACHINE LEARNING MODELS

Black-Box Property Inference Attack In this chapter, we propose a novel black-box property inference attack and relax the key assumptions in prior work. Our prediction normalization technique allows an adversary to perform property inference attacks without knowledge on the target model structure or the training data distribution. In addition, we propose query generation methods that greatly reduce the number of queries required to perform the attack. Our attack achieves over 90% accuracy on three multi-modal datasets and demonstrate the broad risk of property inferences on machine learning models. This chapter is based on joint work with Chenxing Wang, Rui Ye, Carl A. Gunter, Xiaofeng Wang, Michael Backes, and Yang Zhang.

Machine learning (ML) has become increasingly important for a wide range of scientific and industrial applications, such as autonomous driving, face recognition, and natural language processing. However, with the rapid growth of ML technologies, security and privacy issues have emerged. Various attacks have been identified to reveal information about the training data and model structures of a machine learning model. For example, membership inference attacks [101, 102] predict whether an example is in the training dataset of the model, while parameter stealing attacks [17, 103] infer the parameters or hyper-parameters of a model from its predictions.

Recently, a number of studies [14, 8] have demonstrated the substantial risk of property inference attacks. Unlike membership inference attacks, which focus on individual privacy, property inference attacks target the sensitive *global* properties of the training data. Prior studies suggest that property inference attacks could cause severe problems by extracting properties that the model producer do not intend to share. For example, by inferring the hidden distribution of the training data, a competitor may build a more effective model for the same task and get commercial advantage in the business [14]. Moreover, a classifier trained on a log dataset may leak information about security settings of the machines that generated the log [8] and allow the adversary to evade detection or identify vulnerabilities.

Additionally, property inference can also serve good purposes like fairness auditing. Recent research has identified problems caused by under-representation of minority groups in the training dataset [104, 105]. With property inference, a third-party may check whether certain minority group is included in the training dataset or whether the model has biased performance on certain population groups.

Ganju et al. [8] have proposed a white-box property inference attack on Fully-Connected

Neural Networks (FNN). Specifically, they assume the adversary to have access to the model parameters, model structures, and a dataset that shares the same underlying distribution as the training dataset of the model. First, they train some *shadow models* that share the same model structure as the target model. Then, they view each layer of the shadow model as a set and leverage the DeepSet [106] architecture to train a *meta-classifier*. The meta-classifier uses the weights of an FNN as features and predicts the property of the FNN. Finally, they use the meta-classifier to infer the target model's property.

However, the attack relies on some assumptions that limit the scope of scenarios on which the attack can be performed.

First, the attack requires white-box access to the model. Yet, with the increasing competition in the machine learning business, enterprises are less likely to share their model parameters because doing so will allow their competitors to gain commercial advantages in the business. Instead, machine learning models are more likely to be deployed with publicly accessible query interfaces. For example, the Google Vision AI [1] service allows users to query pre-trained image recognition models by uploading images. Meanwhile, the parameters of these models are considered as business secrets and are not shared to public. Although it's possible to reverse-engineer these parameters through the prediction API [17], these attacks often require submitting a large number of queries to the models.

Second, the attack assumes the adversary to know the *exact* structure of the target model. Similar to model parameters, the structures of industrial machine learning models are often considered as business secrets and rarely shared publicly. Although it might be possible to guess the type of model being deployed, knowing the *exact* model structure is almost impossible for a large range ML applications. In addition, limited by the node permutation technique, the attack only works on FNN and is not applicable to other model structures.

Third, to ensure that the shadow models have similar performance as the target model, the prior attack assumes the adversary to have access to a dataset that come from the same distribution as the target model's training dataset. This is a rather strong assumption because because datasets collected from different sources often have slightly different distributions. For example, suppose the target model is trained on a face-image dataset. Although there are several publicly available face-image datasets, they usually contain images of different individuals and therefore do not share the same distribution. Yet, it remains unclear whether the attack could still succeed if the shadow models are trained on different datasets. Additionally, sharing of sensitive data such as medical records are strictly regulated, so it is challenging to obtain any real data that share the same format as the target training data.

[1] https://cloud.google.com/vision/

In such cases, the prior attack would no longer work.

Our Contributions. In this chapter, we propose novel black-box property inference attacks that broaden the scope of property inference attacks by relaxing the key assumptions relied by the prior attack. Specifically, we address three technical challenges that limit the scope of the prior attack: (C1) we need to generate a small number of queries on which the target model's predictions are sufficient to reveal the sensitive property; (C2) the meta-classifier should have good transferability between the shadow models and the target model, even when the models have different structures and training distributions; (C3) when real data is not accessible, synthetic data should be generated with only limited knowledge about the target model's training distribution.

First, we perform property inference attacks under the assumption that the adversary can only interact with the target model by submitting a *limited* number of queries to the model's prediction API (named a *black-box adversary*). Since there is a cost associated with querying a ML application, the number of queries should be relatively small. This limitation reduces the amount of information that could be extracted from the target model. To address the challenge (C1), we design three novel strategies to generate queries: random sampling, query selection, and query crafting. We compare the performance of the three strategies with varying number of queries and demonstrate that all three techniques are effective for property inference. Specifically, using the query crafting technique, we achieve 93% accuracy in inferring the survey year of the Census dataset with a single query to the target model. This result implies that property inference attacks have low cost and impose a much broader threat on real-world ML applications.

Second, we evaluate property inference attacks under an adversary with no background knowledge on the target model (named a *model-agnostic adversary*). To improve the transferabiliy of the meta classifier (addressing C2), we design a *prediction normalization* technique that effectively reduces the difference in predictions between models of varying structures. We demonstrate that the target property of an FNN could be inferred with 98% accuracy using simple logistic regression shadow models on the Insta dataset.

Third, we evaluate property inference attacks under an adversary with no background knowledge on the training distribution (named a *data-agnostic adversary*). Using our prediction normalization technique, we train meta-classifiers that are transferable across models trained on datasets from different distributions. When real data is not accessible, we generate synthetic data based on summary statistics of the training distribution (addressing C3). We demonstrate that property inference attacks could achieve high accuracy even with poor quality synthetic data. Specifically, on the Census dataset, when the shadow models

	Model Type	Parameters	Structure	Distribution
White-Box [8]	FNN	✓	✓	✓
Black-Box	Any	-	✓	✓
Model-Agnostic	Any	-	-	✓
Data-Agnostic	Any	-	✓	-

Table 5.1: Comparison between different threat models.

trained on the synthetic data have only 68.3% accuracy (while the target models have more than 80% accuracy), the attack still achieves 99.4% accuracy. Therefore, protections on the training data do not help prevent property inference attacks.

We evaluate our attacks on three multi-modal datasets: a census dataset (Census [107]), an image dataset (CelebA [108]), and a location dataset (Insta). Our attacks achieve high accuracy on all three datasets under three different adversarial setups (i.e. black-box adversaries, model-agnostic adversaries, and data-agnostic adversaries).

Our contributions can be summarized as follows:

1. We substantially relax the adversarial assumptions of property inference attacks, including the assumption of white-box access, knowledge on the target model structure, and knowledge on the training data distribution.

2. We broaden the scope of property inference attacks by designing attacks that work for more than just FNN models.

3. We design prediction normalization technique that considerably improves the transferability of meta-classifiers.

4. We reduce the cost of black-box property inference attacks with novel query selection and query crafting algorithms.

5. We evaluate property inference attacks under three different adversarial setups and on three multi-modal datasets. Our evaluation results demonstrate the broad threat of property inference on different real-life applications.

Organization. The chapter is organized into 5.4 sections. In Section 5.1.1, we formalize the problem and introduce the threat models. In Section 5.2, we present the attack methodology. In Section 5.3, evaluate our attacks on three different datasets. We end with sections on related work and conclusions.

5.1 PROBLEM STATEMENT

5.1.1 Property Inference on ML Models

Suppose D is a dataset of labeled records. We assume that a machine learning algorithm $\mathcal{A} : D \mapsto f_\theta$ trains a classification model f that predicts the label of each record in D. Specifically,

$$f_\theta : \mathbf{x} \mapsto \mathbf{y}, \tag{5.1}$$

where θ represents the model's parameters, $\mathbf{x} \in X^d$ represents a record of d dimensions, and $\mathbf{y} = (y_1, y_2, ..., y_k)$ is a vector of predicted probability for each class labels. Hence, for each class label $i \in \mathbb{Z}^+$ and $i \leq k$, we have $0 \leq y_i \leq 1$, where k is the number of class labels.

The *property* of f_θ could refer to any sensitive global information related to the training dataset D. Specifically, we define the following *property function*

$$P : f_\theta \mapsto p \quad (p \in \mathbb{Z}^+, p \leq m), \tag{5.2}$$

which maps a machine learning model to a positive integer representing its property. m is the number of possible properties. For example, suppose D is a census dataset, and the property is the education level of individuals in D, which takes one of the following three values: high school, undergraduate, and graduate. We can define the following property function P:

1. If individuals in D have high school education level, $P(f_\theta) = 0$;
2. If individuals in D have undergraduate education level, $P(f_\theta) = 1$;
3. If individuals in D have graduate education level, $P(f_\theta) = 2$.

In a property inference attack, the goal of the adversary is to infer a property $p = P(f_\theta)$ of a classification model f_θ. We call f_θ the *target model* and p the *target property*.

5.1.2 Threat Models

Karan et al [8] has demonstrated the possibility of property inference attacks under the white-box setting. However, the white-box attack relies on some strong assumptions which reduce the scope of the attack. In this chapter, we study property inference attacks under three more practical threat models. By relaxing the key assumptions made in the prior attack, we show that property inference attacks can be performed under a broader range of scenarios.

Black-Box Adversary. The prior attack relies on the assumption that the adversary has white-box access to the target model. Specifically, given a target model f_θ, the adversary can directly access the model parameters θ. However, in practice, instead of sharing the model parameters, the model owner is more likely to allow users to interact with the model through a prediction API.

We assume that a *black-box adversary* can only interact with the target model f_θ through its prediction API. Specifically, the adversary can submit a set of unlabeled records $\{\mathbf{x}_i \mid \mathbf{x}_i \in X^d, i \in \mathbb{Z}^+, i \leq n\}$ to the target model, which returns a prediction vector $f_\theta(\mathbf{x}_i)$ for each record $\mathbf{x}_i$. We call $\mathbf{x}_i$ a *query* to the target model.

Ideally, a black-box adversary should be allowed to submit unlimited number of queries. However, in practice, each query incurs some time and financial cost on the adversary. Therefore, we assume that the adversary can submit at most n queries.

Model-Agnostic Adversary. Although a black-box adversary does not have access to the model parameters, she may have some background knowledge about the target model, such as the training algorithm $\mathcal{A}$ and the model structure [101]. On the contrary, a *model-agnostic adversary* has no background knowledge on the target model, except for its input and output format. For example, suppose the target model is a classifier that predicts the salary class (e.g. > 50K or ≤ 50K) of an individual based on his/her demographic information. A model-agnostic adversary only knows the model's input features (i.e. a vector of demographic information) and the classification task (i.e. predicting the salary class of an individual). However, she has no knowledge about the training algorithm or the model structure.

Data-Agnostic Adversary. Another important assumption relied by the prior attack is that the adversary has access to a dataset D' which is sampled from the *same* distribution as the target model's training dataset D. To relax this assumption, we assume that a *data-agnostic adversary* has no knowledge about the distribution from which the training dataset D is sampled. Instead, she may have access to a dataset D', which is sampled from a *different* data distribution. For example, if D is a face-image dataset, D' could be a dataset containing faces of a different group of individuals; if D is a dataset of checkin locations for users in New York, D' could be a dataset of checkin locations for users in a different city. If data from a different distribution is not available, the adversary may have access to some summary statistics about the training data, such as the marginal distribution of features in each class. Compared to real data, this information is much easier to obtain because it brings less privacy concern. For example, sharing aggregate statistics is not covered by the HIPPA privacy rule [109], which regulates the sharing of medical data.

5.2 METHODOLOGY

In this section, we introduce our attack methodologies under three different adversarial models: the black-box adversary, the model-agnostic adversary, and the data-agnostic adversary.

5.2.1 Black-Box Adversary

Our black-box attack consists of four major steps: training shadow models, generating queries, training the meta-classifier, and attacking the target model. Below, we briefly introduce each steps of the attack.

Training Shadow Models. Shadow models have been widely used in different attacks on machine learning models [101, 110, 8]. They are models that share similar structures and training data distributions as the target model. Therefore, an adversary can use shadow models to imitate the performance of the target model and generate training data for the attack.

Assume that the adversary has access to the training algorithm $\mathcal{A}$ and a shadow dataset D_{shadow} that comes from the same data distribution as the training dataset D of the target model. Suppose the target models have m possible properties $p_1, p_2, ..., p_m$. By sampling from the shadow dataset D_{shadow}, for each property p_i, we obtain s subsets that follow the property p_i, denoted as $D^{(i,j)}_{\text{shadow}}$, where $j \in \mathbb{Z}^+$ and $j \leq s$.For each subset, we train a shadow model

$$f^{(i,j)}_{\text{shadow}} = \mathcal{A}\left(D^{(i,j)}_{\text{shadow}}\right). \tag{5.3}$$

According the definition of the property function P (Section 5.1.1), we have $P(f^{(i,j)}_{\text{shadow}}) = p_i$.

We use $\mathcal{F}_{\text{shadow}}$ to denote the set of shadow models

$$\mathcal{F}_{\text{shadow}} = \{f^{(i,j)}_{\text{shadow}} \mid i, j \in \mathbb{Z}^+, i \leq m, j \leq s\}. \tag{5.4}$$

Generating Queries. In a black-box attack, the adversary does not have access to the target model's parameters θ. However, she could submit a query (i.e. an unlabeled record) $\mathbf{x}$ and obtain the model's prediction $\mathbf{y} = f_\theta(\mathbf{x})$, which is a vector of predicted probabilities for all the classes. If two classification models share the same property, their predictions on $\mathbf{x}$ should be similar. Therefore, the prediction vectors could be used as features for black-box property inferences.

To extract more information from the target model, an adversary could generate a set of n queries:

$$Q = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}. \tag{5.5}$$

Then, the *property inference feature* of a model f could be represented as a concatenation of the prediction vectors:

$$\mathbf{o} = (f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_n)). \tag{5.6}$$

$\mathbf{o}$ serves as the feature vector for the meta-classifier. Increasing the number of queries n would allow the adversary to extract more information from the target model. However, each query may lead to additional cost for the adversary. On the one hand, there is a cost associated with submitting queries to ML applications. On the other hand, with more features, the adversary needs to train more shadow models and build more complex meta-classifiers. Therefore, to reduce attack cost, we perform property inference with a small number of queries. Specifically, we design three methods for query generation: random sampling, query selection, and query crafting. These methods are introduced in Section 5.2.4.

Training the Meta-Classifier. Given a set of queries Q and a set of shadow models $\mathcal{F}_{\text{shadow}}$, we train a meta-classifier g to predict the sensitive property. For each shadow model $f^{(i,j)}_{\text{shadow}} \in \mathcal{F}_{\text{shadow}}$ with property p_i, we obtain a property inference feature

$$\mathbf{o}^{(i,j)} = \left(f^{(i,j)}(\mathbf{x}_1), f^{(i,j)}(\mathbf{x}_2), \ldots, f^{(i,j)}(\mathbf{x}_n)\right). \tag{5.7}$$

The pair $(\mathbf{o}^{(i,j)}, p_i)$ forms a training record for the meta-classifier. The meta-classifier g is trained to predict the sensitive property p_i based on the property inference feature $\mathbf{o}^{(i,j)}$ by minimizing its classification loss over all the shadow models.

Attacking the Target Model With the meta-classifier g, an adversary could infer the sensitive property of any target model based on its prediction on the query set Q. Suppose $\mathbf{o}_{\text{target}}$ is the property inference feature generated by querying the target model f_{target}, the prediction $g(\mathbf{o}_{\text{target}})$ reveals the sensitive property of f_{target}.

5.2.2 Model-Agnostic Adversary

A model-agnostic adversary has no background knowledge on the target model, except for its input and output format. Therefore, the shadow models $\mathcal{F}_{\text{shadow}}$ do not share the same structure as the target model f_{target}. This difference may lead to the discrepancy between the

predictions of the shadow models and the target model. Consequently, a meta-classifier that is trained on the predictions of shadow models could have poor performance on the target model.

Prediction Normalization. The key to address the challenge of a model-agnostic attack is to reduce the difference between the property inference features of shadow models and target models.

We observe that, although models with different structures often give distinct predicted probabilities on the same query, the relationship between the predictions on different queries remain the same. Therefore, we apply *prediction normalization* technique to reduce the difference in the mean and the variance of of the predictions. Specifically, given a property inference feature $\mathbf{o}$ of length n, let o_i represent the i-th element of $\mathbf{o}$. First, we calculate the average prediction:

$$\bar{o} = \frac{1}{n}\sum_{i=1}^{n} o_i, \tag{5.8}$$

and the standard deviation:

$$\sigma_o = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(o_i - \bar{o})^2}. \tag{5.9}$$

Then, we obtain the normalized property inference feature

$$\hat{\mathbf{o}} = \frac{\mathbf{o} - \bar{o}}{\sigma_o}. \tag{5.10}$$

Although prediction normalization is similar to feature normalization in machine learning, the dimension on which the normalization is performed is different. In machine learning, normalization is often performed across all the records to ensure that the features share the same mean and variance. On the contrary, in prediction normalization, the normalization is performed across all the property inference features (i.e. a model's predictions on different queries) to ensure that all the records have the same feature mean and feature variance. This normalization process reduces the difference between the training records, generated by the shadow models, and the testing records, generated by the target models. Consequently, the meta-classifier has better transferability between the shadow models and the target models.

5.2.3 Data-Agnostic Adversary

We assume that a data-agnostic adversary has no access to data that come from the same distribution as the target training data. Specifically, we consider two possible scenarios: (1)

the adversary has access to the same type of data coming from a different distribution; (2) the adversary do not have access to any real data, but may know some summary statistics of the training data.

First, we consider an adversary with access to data from a different distribution. For example, if the target training set contains images of faces, the adversary may have access to face images of different individuals, which is much easier to obtain in practice. Another example is that an adversary could use location data of city A to attack models trained on location data of city B, if collecting data from the same city as the target model is impractical.

We observe that training shadow models on data from a different distribution has a similar effect as training shadow models with a different model structure—the predictions may differ between shadow models and target models, yet the relationship between the predictions on different queries are often similar. Therefore, we apply prediction normalization to reduce the difference between predictions of shadow models and target models.

Second, when no real data is available, synthetic data need to be generated based on the limited knowledge about the target model's training data. It is noteworthy that, under this assumption, it is not feasible to generate high-quality synthetic data that share the same distribution as the target training data. Instead, we design a simple synthetic data generation method and demonstrate that high-accuracy property inference attacks are possible even with shadow models trained on low-quality synthetic data.

Specifically, we assume that the adversary has access to the marginal distribution of each feature given the class label and sensitive property, and the synthetic data are generated by randomly sampling based on the marginal distribution. We first generate a class label and then independently generate each feature based on its conditional probability given the class label.

5.2.4 Query Generation

The key to query generation is to find out queries on which the model's prediction has high correlation with the sensitive property. In this section, we introduce three methods for query generation: random sampling, query selection, and query crafting.

Random Sampling. *Random sampling* is the process of random choosing queries from a set of real or synthetic data available to the adversary. Since each query extracts some different information about the target model, with sufficiently large number of queries, the adversary could gather enough information to infer the sensitive property.

However, the performance of this approach is limited by the number of queries an adversary could submit to the target model. When an adversary is only allowed to query the model

a few times, the randomly sampled queries may not extract enough information from the target model and the attack accuracy would drop.

Query Selection. To reduce attack cost, we need to select queries that are highly correlated to the sensitive property we want to predict. This correlation could be measured by the mutual information between the the sensitive property and the shadow models' predictions on the query.

Suppose X_i is the set of predictions on query $\mathbf{x}_i$, and Y is the set of sensitive properties. Then X_i is continuous and Y is discrete. The mutual information between X_i and Y is defined as

$$\mathrm{MI}(X_i; Y) = \sum_{y \in \mathcal{Y}} \int_{x \in \mathcal{X}} p_{(X,Y)}(x, y) \log \left(\frac{p_{(X,Y)}(x, y)}{p_X(x) p_Y(y)} \right) dx, \tag{5.11}$$

where $\mathcal{X}, \mathcal{Y}$ is the range of X_i and Y respectively, while p_X, p_Y, and $p_{(X,Y)}$ are the probability density functions. We use the MI estimator based on k-nearest neighbor [111] implemented in `sklearn` [2] to calculate $\mathrm{MI}(X_i; Y)$.

The query selection process contains two steps. First, an adversary estimates the mutual information between each query and the sensitive property using the shadow models. Then, she select the top n queries with the highest mutual information and submit them to the target model. Since the first step is performed locally it incurs no extra cost to the adversary, and the attack cost is determined by the number of selected queries.

Query Crafting. Although query selection allows an adversary to pick the most correlated queries, it only selects from data accessible by the adversary. However, in practice, a query could be *any* vector feature space. Therefore, we design query crafting method to generate queries that maximize the performance of a property inference attack. .

We use g_θ to represent the meta-classifier with parameters θ and $\mathbf{q}$ to represent the query. Suppose $\mathcal{L}(\theta, \mathbf{q})$ is the meta-classifier's loss on the shadow models. We adapt the algorithm in [103] to minimize $\mathcal{L}$.

First, we update the parameters θ of the shadow models by descending the gradients of $\mathcal{L}$ on θ:

$$\theta \leftarrow \arg\min_{\theta} \mathcal{L}(\theta, \mathbf{q}). \tag{5.12}$$

Then, we modify the query $\mathbf{q}$ by descending the gradients of $\mathcal{L}$ on $\mathbf{q}$:

$$\mathbf{q} \leftarrow \arg\min_{\mathbf{q}} \mathcal{L}(\theta, \mathbf{q}). \tag{5.13}$$

[2] `scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html`

Dataset	Target Property		Classification Task
	P1	P2	
Census	Year 94	Year 95	Salary Class
CelebA	Smiling	Non-Smiling	Gender
Insta	Male	Female	Location Category

Table 5.2: Target properties and classification tasks.

Dataset	Target	Shadow	Shadow (Different Structure)	Shadow (Different Data)
Census	0.801	0.791	0.762	0.683
CelebA	0.814	0.812	0.799	0.844
Insta-London	0.756	0.761	0.687	0.742

Table 5.3: Average test accuracy of target models and shadow models on their classification tasks.

Attack Method	Random Sampling	Query Selection	Query Crafting
# of Queries	100	10	1
Census	0.951	0.975	0.930
CelebA	1.000	0.982	0.943
Insta-London	0.993	0.995	1.000

Table 5.4: Comparison between different query generation methods in black-box attacks.

We repeat steps (5.12) and (5.13) until convergence. Then, we use the meta-classifier g_θ and the crafted query $\mathbf{q}$ to infer the sensitive property of the target model.

5.3 EVALUATION

In this section, we evaluate our attacks on three multi-modal datasets.

5.3.1 Experiment Setup

We first introduce the datasets and models used in our experiments and define the target properties in each dataset. Table 5.2 presents the target properties and classification tasks of each dataset. In each attack, we repeated the attack 10 times and reported the average attack accuracy on the target models among 10 repetitions.

US Census Income Data (Census) The U.S. census income dataset [107] contains 299285 weighted census data extracted from 1994 and 1995 population surveys conducted by U.S. Census Bureau. It contains 40 different attributes related to demographic and employment

information. The classification task for this dataset is to determine the income level for each record. Incomes have been binned at the 50K level, which makes this problem a binary classification problem.

Among the 40 attributes, 7 of them are continuous and 33 of them are nominal. In the prepossessing step, continuous attributes are normalized to $[0, 1]$ and hot-encoding is applied on the nominal attributes. After prepossessing step, we got 410 attributes in our dataset. We constructed an FNN with 3 hidden layers of size 32, 16, 8 respectively. We used ReLU activation function in each layer. In the output layer, we used Softmax function to map the outcome into $[0, 1]$

The target property we tried to infer is the year in which the population survey. It has two possible values "1994" and "1995". We trained 100 different target models on records with attribute "1994" and 100 different target models on records with attribute "1995".

CelebFaces Attributes (CelebA) . The CelebA dataset [108] is a large-scale face attributes dataset with more than 200K celebrity images belonging to about 10K identities. Each image is of size 218×178 and has 40 binary attributes, such as gender, smiling or not, wearing hat or not. Our classification task is to detect the gender of people in the image.

In the prepossessing step, we reshaped each image as a vector and normalized it in to $[0, 1]$. Then we trained an FNN with 3 hidden layers of size 640, 32, 8 respectively. We used ReLU activation function in each layer. In the output layer, we used Softmax function to map the outcome into $[0, 1]$

The target property we tried to infer is whether individuals in the images are smiling or not. We first separated all the images into two image sets based on the target property. Then, we trained 100 target models with "smiling" images and 100 target models with "non-smiling" images. Each target model was trained on 2000 randomly selected images.

Insta dataset . The Insta dataset contains check-in information of over 300K locations in London and Los Angeles (LA). Each check-in record contains check-in time and check-in user information (including gender, age, and user id). The locations are classified into 9 different categories such as park, restaurant, and hotel. We selected all the locations from the restaurant category and the park category to set up a binary classification task. We deleted all the locations with smaller than 5 check-in records. We constructed training features based on check-in time. Each feature vector has 24 values, and each value represents the percentage of check-in records in the corresponding hour. We built an FNN with 2 hidden layers of size 30 and 10 respectively. We used ReLU activation function in each layer. In the output layer, we used Softmax function to map the outcome into $[0, 1]$. The model predicts the location category based on check-in time.

Attack Method	Random Sampling	Query Selection	Query Selection w/ Normalization
# of Queries	100	10	10
Census	0.500	0.500	**0.834**
CelebA	0.970	0.990	**0.990**
Insta-London	0.595	0.968	**0.980**

Table 5.5: Performance of model-agnostic attacks (best results are bolded).

We used check-in records in London to build the target model. The target property is the gender of check-in users in the training dataset. We separated our check-in records into two sets, one of them includes all the check-ins of male users and the other includes all the check-ins of female user. Then, we trained 100 target models with check-ins of male users and 100 target models with check-ins of female users.

5.3.2 Black-Box Adversary

In black-box attacks, we trained shadow models with the same structure and training data distribution as the target models. Table 5.3 presents the average test accuracy of target models and shadow models on their classification tasks. The models have similar test accuracy, which allows the meta-classifier to easily transfer between shadow models and target models.

We performed black-box property inference attacks with three different query generation methods: random sampling, query selection, and query crafting. Table 5.4 presents the performance of the attacks. All three methods achieve over 90% attack accuracy. Additionally, query selection and query crafting substantially reduce the number of queries to the target model. Specifically, query selection with 10 queries achieve the same level of performance as random sampling with 100 queries. What stand out the most is the performance of query crafting, which only requires a single query to the target model. This result indicates that property inference attacks have low attack cost and are hard to detect. It is challenging to prevent the attack based on query monitoring since it only takes one query to infer the sensitive property.

5.3.3 Model-Agnostic Adversary

In model-agnostic attacks, the adversary do not know the structure of the target model. Therefore, we trained shadows models that have different structures as the target model.

Attack Method	Random Sampling	Query Selection	Query Selection w/ Normalization
# of Queries	100	10	10
Census	0.897	0.500	**0.994**
CelebA	0.923	0.944	**0.982**
Insta-London	0.556	0.840	**0.905**

Table 5.6: Performance of data-agnostic attacks (best results are bolded).

We used logistic regression on Census and Insta-London datasets. For the CelebA dataset, we trained FNNs with different structures because logistic regression is not commonly used on image data. Specifically, we trained FNNs with 4 hidden layers of size 640, 128, 16, 4 respectively, while target models have 3 hidden layers with size 640, 32, 8 respectively.

As shown in Table 5.3, when shadow models and target models have different structures, there is a gap in their average test accuracy. This gap is determined by how much the structure of the shadow models is different from the structures of the target models. For example, on the Insta-London dataset, the test accuracy of shadow models are around 10% lower than the accuracy of target models, while on the CelebA dataset the difference is much smaller. This difference makes property inference attacks more challenging because the patterns in the predictions of shadow models may not transfer to the predictions of target models.

Indeed, Table 5.5 shows that. when there is a significant difference between the accuracy of target models and shadow models, the model agnostic performance of property inference attacks drop significantly when prediction normalization is not used. Specifically, on the Census dataset, the attack performance are equivalent to random guessing without prediction normalization.

Table 5.5 also highlights the effectiveness of prediction normalization in maintaining a high accuracy in model-agnostic attacks. Attacks with prediction normalization achieve over 80% accuracy on all three datasets. Particularly, the performance gain on the Census dataset is over 30%.

5.3.4 Data-Agnostic Adversary

To evaluate the performance of data agnostic attacks, we considered three different assumptions about the adversary knowledge shown in Table 5.7. For attacks on the Census dataset, we assumed that the adversary does not have access to any real data and trained shadow models on synthetic data generated from the marginal distributions of features. For attacks

Dataset	Adversary Knowledge
Census	Marginal distribution of features
CelebA	Image of different individuals
Insta-London	Data from a different city (Insta-LA)

Table 5.7: Knowledge of data-agnostic adversaries.

on the CelebA dataset, we assumed the adversary to have access to face images of a different group of individuals *not* included in the training dataset of the target model. For attacks on the Insta-London dataset, we assumed the adversary to have access to the Insta-LA dataset, which is collected from check-in records in a different city.

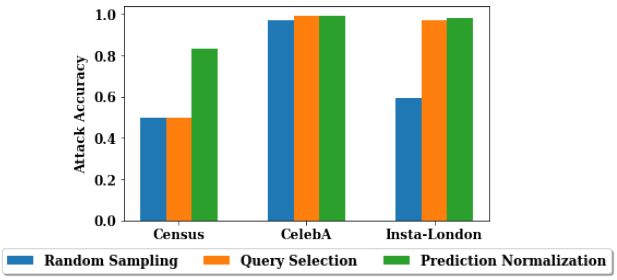
Table 5.3 presents the test accuracy of target models and shadow models. To present a fair comparison between the two groups of models, we evaluated the test accuracy on real test data sampled from the same distribution the training data of the target models. It is noteworthy that this information was used only for the comparison and was *not* used when performing property inference attacks. In practice, the adversary does not need to know the test accuracy of shadow models on real data.

It is apparent from Table 5.3 that, on the Census dataset, shadow models in data-agnostic attacks have a much lower testing accuracy compared to target models. Specifically, shadow models trained on synthetic census data have an average test accuracy of 68.3%, which is 0.118 lower than the average test accuracy of target models. This difference indicates the challenge in performing data-agnostic property inference attacks.

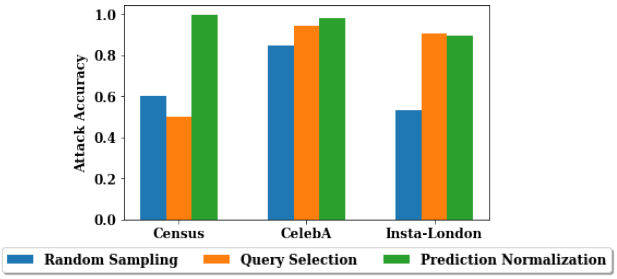
However, despite of the difficulty in obtaining high-quality shadow models, data-agnostic property inference attacks still have a high success rate. As shown in Table 5.6, with prediction normalization, the attack accuracy reaches 99.4% on Census dataset, which demonstrates that there is a severe risk in property inference even when the adversary does *not* have access to any real data.

5.3.5 Results Analysis

Effectiveness of Prediction Normalization and Query Selection. To further understand the effectiveness of prediction normalization and query selection, we compared the attack performance with 10 queries to the target model. Figure 5.1 shows the attack performance under different attack strategies (i.e., random sampling, query selection, and query selection with prediction normalization). In almost all the settings, attacks with query selection and prediction normalization achieve the highest accuracy, and the attack accuracy is always higher than 80%. Interestingly, the effectiveness of query selection varies among



(a) **Model Agnostic Attacks.**



(b) **Data Agnostic Attacks.**

Figure 5.1: **Attack performance with 10 queries.** The y-axis shows the attack accuracy, while the x-axis shows the dataset on which the attack is performed. We compared the attack performance under three strategies: query generation using random sampling , query selection, and query selection with prediction normalization. Attacks with prediction normalization have the highest accuracy for both model-agnostic adversary and data-agnostic adversary.

datasets. On the Census dataset, query selection brings little performance gain compared to using random sampling. However, on the Insta dataset, query selection increases the attack accuracy by more than 0.3. One possible explanation for this scenario is that, there is a larger variation among records in the Insta dataset (i.e. locations) compared to records in the Census dataset (i.e. individuals). Therefore, selecting the right records to query is more important when attacking the Insta dataset.

Difference between Shadow Models and Target Models. Figure 5.2 presents the average test accuracy of target models and shadow models on their classification tasks. Theoretically, when there is a larger difference between the test accuracy of shadow models and target models, it is more challenging to achieve high accuracy in property inference attacks. Our attack results with random sampled queries confirm with this hypothesis. However, attack results with prediction normalization demonstrate that property inference attacks can achieve high accuracy even with shadow models whose performances are significantly different from the target models. This result demonstrates a broader and more severe risk of

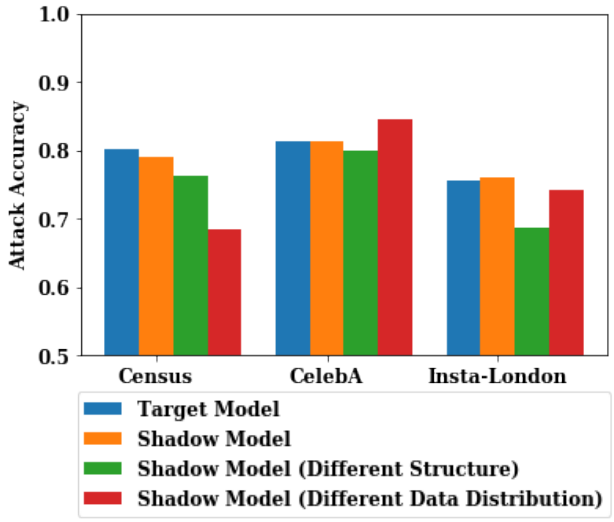


Figure 5.2: **Average test accuracy of target models and shadow models on their classification tasks.** We compared the test accuracy between shadow models and target models in different attacks. The x-axis represents the datasets, and the y-axis shows the test accuracy of models on their classification tasks.

property inferences on machine learning models.

5.4 CONCLUSION

In this chapter, we proposed a novel black-box property inference attack against machine learning models. Specifically, we designed three query generation methods to extract information for property inference. Our query selection and query crafting technique could greatly reduce the number of queries required for the attack. Additionally, we developed prediction normalization technique that allows an adversary to perform property inference attacks without knowledge on the target model structure or the training data distribution. We showed that our attacks are effective at inferring various data properties on multiple real-world datasets.

CHAPTER 6: A HYPOTHESIS TESTING APPROACH TO SHARING LOGS WITH CONFIDENCE

In this chapter, we introduce a game-based definition of the risk of exposing sensitive information through released logs. We propose log indistinguishability*, a property that is met only when the logs leak little information about the protected sensitive attributes. We design an end-to-end framework that allows a user to identify risk of information leakage in logs, to protect the exposure with log redaction and obfuscation, and to release the logs with a much lower risk of exposing the sensitive attribute. Our framework contains a set of statistical tests to identify violations of the log indistinguishability property and a variety of obfuscation methods to prevent the leakage of sensitive information. The framework views the log-generating process as a black-box and can therefore be applied to different systems and processes. We perform case studies on two different types of log datasets: Spark event log and hardware counters. We show that our framework is effective in preventing the leakage of the sensitive attribute with a reasonable testing time and an acceptable utility loss in logs. This chapter is based on joint work with Le Xu and Carl A. Gunter [112].*

Logs generated by systems and applications contain a wide variety of information such as timestamps, the size of input and output files, and CPU utilization. This information plays an important role in scientific research and analysis of production systems in industry. Although some of these analyses can be done internally, there has been a growing need to share logs with external analysts. For example, researchers rely on logs from real-life production systems to understand the workload and performance of these systems, and large companies hope to share this information so that they can guide academic work to be more relevant to their technical challenges [9]. On the other hand, small companies often outsource their log analysis process to third-party service providers such as Anomaly (`anomaly.io`) and Anodot (`anodot.com`) to benefit from a larger log-database and cutting-edge technologies.

Despite the increasing need for log sharing, companies are often unwilling to share their logs due to concerns about information exposure. Indeed, logs have the potential of revealing sensitive information about the system or program that generated it, ranging from applications, algorithms, to software and hardware configurations. For example, releasing traces of real-life production workloads may result in the leakage of information about new products if the prototypes of these products are using the same infrastructure [9].

Some sensitive information is directly saved into logs and can be protected by sanitization or anonymization. For example, several attempts have been made to anonymize IP addresses in network traces [43, 44]. However, log anonymization and sanitization is not enough to

prevent the leakage of all the sensitive information. Since the metrics stored in logs are highly correlated with various properties of the system and hardware, seemingly nonsensitive metrics may reveal sensitive information. For instance, it is possible to infer information about a physical machine (e.g. the amount of RAM, the number of cores) if both workload information and performance metrics are released.

Differential privacy [6] has been shown to be effective in preventing side-channel leakages from timing and message sizes [113], but these studies only cover a portion of the information that is included in a log file. Meanwhile, achieving differential privacy on high-dimensional complex data with a reasonable amount of noise remains an open research problem [114].

In contrast to the extensive research on log anonymization, there is a lack of a systematic understanding of the sensitive information that can be indirectly inferred from a log file. Yet, this indirect leakage has been a great concern of companies in terms of releasing production logs. To address this concern, researchers at Google proposed several obfuscation techniques [9], including sampling, scaling, and aggregation, to prevent the information leakage. However, there are two key limitations in these techniques. First, the obfuscation techniques are designed specifically to protect the traces from Google production clusters. The effectiveness of these techniques need to be generalized to different protection criterion, different systems, and different types of log files. Second, due to the lack of a clear protection criterion, there is no quantitative analysis of the effectiveness of the obfuscation techniques. Consequently, it is challenging for a user to understand the goal of each obfuscation techniques and to adapt them for their own systems.

Our Contributions. This chapter aims to provide a framework that enables general-purpose log sharing under user-specified protection requirements. This framework consists of three major components: protection specification, indistinguishability tests, and log obfuscation.

First, we formalize the risk of revealing sensitive information during log sharing under the definition of *log indistinguishability*. We then enable users (log producers) to specify some sensitive attributes to be protected. We model the process of inferring the sensitive attributes from logs as a distinguishing game between the adversary and the user. The log indistinguishability property is met only if the adversary gets little or no advantage over random guessing in the distinguishing game. Log indistinguishability provides a formal protection criterion for our log-sharing framework. The remainder of the framework is designed around checking and ensuring this criterion in shared logs.

Second, we propose a set of *indistinguishability tests* to check whether a log file satisfies log indistinguishability. Similar to the randomness tests for Psuedo-Random Number Generators

(PRNG), we use hypothesis tests to identify statistical patterns that may leak the sensitive information in a log file. We design a variety of tests to cover different data types.

Third, we propose log obfuscation techniques to help mitigate the information leakage identified by the indistinguishability tests. We design an iterative process between testing and obfuscation to help users strike a balance between protection of the sensitive information and the information loss incurred by obfuscation.

Finally, we evaluate our log-sharing framework under two case studies on different types of log datasets: Spark event logs and hardware performance counters. We show that our framework is effective in preventing the leakage of the sensitive attribute with a reasonable testing time and an acceptable information loss in logs.

Our contributions can be summarized as follows:

- We introduce a game-based definition of the risk for exposing sensitive information through log sharing.
- We design an end-to-end framework that allows users to identify risk of information leakage in logs, to protect the exposure with log obfuscation, and to release the logs with a much lower risk.
- With two case studies, we show that our framework is effective in preventing the leakage of the sensitive attribute with a reasonable testing time and an acceptable utility loss.

6.1 LOG INDISTINGUISHABILITY

In this section, we formalize the risk of leaking sensitive information through shared logs. First, we introduce the log sharing problem and the adversary model used in this chapter. Then, we propose a game-based definition to describe the risk of leaking the sensitive information. Finally, we provide an overview for our testing-based log obfuscation framework.

6.1.1 Problem Statement

Problem Setup. Suppose P is a log-generating process, and l is the log file produced by P. We study what information can be inferred about P by observing l. We assume l is parsed into a multidimensional time sequence vector consisting of numerical and categorical data. There have been extensive prior studies on parsing unstructured logs into structured sequential data [115, 116].

The sensitive information of P can refer to any information that is related to the computation process and not directly stored in the log file. This information may include software and hardware configurations, information about the physical machines (e.g. number of cores, the amount of RAM), and workload information such as algorithms and hyperparameters. In practice, the information that needs to be protected varies among applications. Therefore, we allow the users to specify the sensitive information that needs to be protected. The remaining information about P is considered to be nonsensitive and safe to release. The sensitive information can be a combination of different attributes of P. However, for simplicity, we view all the sensitive information as a single *sensitive attribute*, denoted by X, and X can be a vector of different configurations. Let $C = \{x_i \mid i \in \mathbb{Z}^+, i \leq M\}$ be a set of potential values for X known by the adversary. We call C the *candidate set* of X.

Adversary Model. We consider an external adversary that does *not* collocate with the target program P and has no control over P. The adversary can only infer information by analyzing the log files shared by the users, and users can sanitize or remove any sensitive information before sharing the log files. We also assume that the adversary has access to all the nonsensitive information about P and can reproduce the experiment on similar hardware and software environments.

An Example. Suppose P is the process of training a deep learning model on a Spark [117] system, and l is a parsed log file produced during the training process. The owner wants to share the log file but is concerned that it may reveal the number of cores of the physical machines used to train the models. In this case, the number of cores is the sensitive attribute X while other information, such as the training algorithm and software configurations, is nonsensitive.

We assume that the adversary knows a set of possible values for the number of cores (e.g. $C = \{1, 2, 4, 8\}$). In addition, the adversary has access to a variety of machines with different number of cores, the same system environment as the user (i.e. the Spark system), and the training algorithm used by the user (program P). The goal of the adversary is to infer the number of cores of the machines. To gather information for the inference, the adversary can generate multiple log files (l) by running the process P on machines with different number of cores. This strong adversary model allows us to provide protection against adversaries with different background knowledge in practice.

6.1.2 Log Indistinguishability

In this section, we propose a game-based definition to formalize the inference process. Based on this definition, we discuss the requirements for preventing the leakage of the sensitive information.

The Distinguishing Game. Given a process P and a candidate set C, we model the problem of inferring the sensitive attribute as the following distinguishing game between the user and the adversary:

1. The user picks a value $x \in C$ uniformly at random and generates a log file l.
2. The user invokes the adversary to obtain a guess $x' = \mathcal{A}(l, C)$.
3. The attack succeeds if $x' = x$. Otherwise, the attack fails.

In the distinguishing game, the adversary aims to infer the sensitive attribute that is used to produce the log file l. She has access to the candidate set C, which contains a set of potential values for the sensitive attribute. The adversary obtains a guess $x' \in C$ based on some inference strategy $\mathcal{A}$. The attack succeeds only if $x' = x$.

Log Indistinguishability. A program P is γ-*log indistinguishable* on C if, for all $C_{\text{pair}} \subseteq C$ with $|C_{\text{pair}}| = 2$, no adversary can succeed the above distinguishing game with probability greater than $(1 + \gamma)/2$ on C_{pair}.

Log indistinguishability guarantees that no adversary can get a significant advantage over a random guess in inferring the sensitive attribute among any pair of possible values. Therefore, the protection holds even when the adversary is able to eliminate some possible values in C. Specifically, when $\gamma = 0$, no inference strategy outperforms random guessing in winning the distinguishing game, indicating that the log file l leaks no information about the sensitive attribute.

6.1.3 Framework Overview

Figure 6.1(a) presents an overview of the log-sharing process, which is inspired by software testing process. In software testing, a set of test modules are designed to identify violations of user-specified requirements. Developers use the test modules to identify bugs in the software. The software is ready to be deployed only if all tests are passed. This process may involve several iterations between implementation and testing. Although software testing cannot

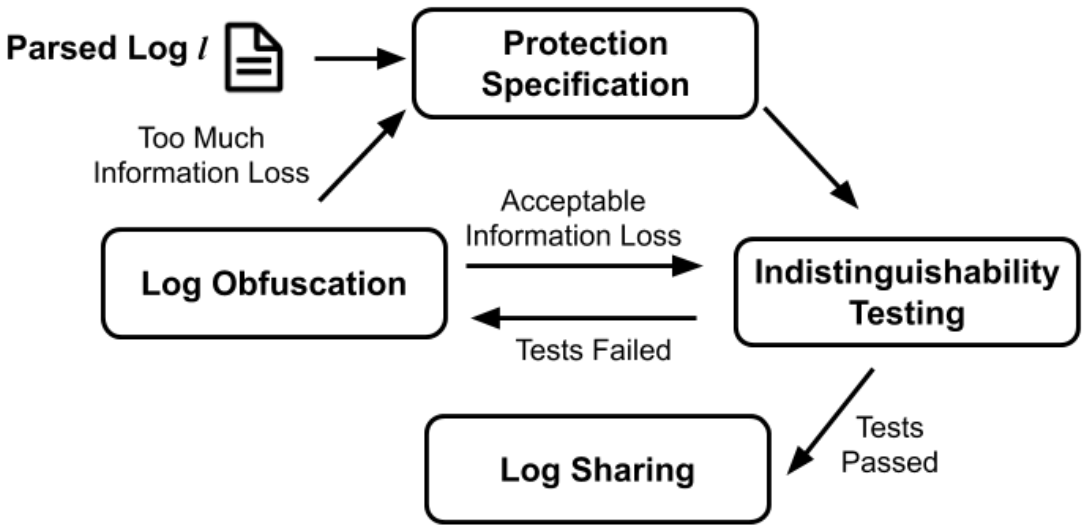


Figure 6.1: Framework Overview

guarantee the software to be bug-free, it is the major approach to detect implementation errors and software defects.

Similarly, our testing-based framework aims to provide users an efficient and effective approach to the identification and protection of information leakage in log files. The framework checks an input log file against a protection criterion specified by the user. If the log file fails to meet the criterion, the framework would provide the user with a set of applicable obfuscation methods. The user decides whether the obfuscated log incurs too much information loss and updates the protection specification accordingly. The log can be confidently shared when it passes all the tests. Similar to software testing, indistinguishability testing cannot guarantee that shared log leaks no information. However, it is a practical method to provide confidence that sharing the log has low risk. Moreover, the framework views the log-generating process as a black-box and can therefore be applied to different systems and processes. Below, we briefly introduce each stage in the log-sharing framework:

1. **Protection Specification.** A user specifies the sensitive attribute X and the candidate set C for a log-generating process P. The goal of the protection is to have P achieve γ-log indistinguishability on C given a small constant γ.

2. **Indistinguishability Testing (Section 6.2).** A set of indistinguishability tests are performed to identify violations of the protection criterion.

3. **Log Obfuscation (Section 6.3).** The log needs to be obfuscated if it fails any of the indistinguishability tests. We study a range of existing obfuscation methods and propose novel methods to protect the sensitive attribute.

4. **Log Sharing.** When all the indistinguishability tests are passed, the user can share the log under a much lower risk of exposing the sensitive attribute.

6.2 INDISTINGUISHABILITY TESTS

In this section, we introduce a set of indistinguishability tests to identify potential violations of γ-indistinguishability. First, we introduce some general principals for designing indistinguishability tests. Then, we propose four types of tests that cover different information in logs. Finally, we introduce methods to interpret and combine the results from different tests.

6.2.1 A Testing-Based Approach

Challenges in Obtaining Theoretical Guarantee. To obtain the theoretical guarantee of γ-log indistinguishability, one could either (i) prove that all the metrics influenced by the sensitive attribute are removed or (ii) show that the obfuscation methods have effectively hidden all the influence of the sensitive attribute. However, both approaches are challenging in practice.

For the first approach, it is difficult to identify *all* metrics that are influenced by the sensitive attribute. Moreover, there are complex correlations between the capacity of physical machines, software and hardware configurations, and performance metrics. Therefore, obtaining a theoretical proof on the influence of a software or hardware setting is generally impractical. Theoretical analysis of a system usually relies on the assumptions that the data obtained from the system follow a known distribution (e.g. Gaussian distribution) [118]. However, many studies suggest that these assumptions do not hold in practice [119, 120].

For the second approach, most obfuscation methods suffer from the lack of certainty—one cannot prove the effectiveness of commonly used obfuscation mechanisms [9]. Meanwhile, obfuscation methods that do provide certainty for protection, such as differential privacy, often reduces the accuracy in log analysis by adding too much noise.

A Testing-Based Approach. Although obtaining theoretical guarantee is challenging, it is feasible to identify patterns in a log file that violate γ-log indistinguishability. In this section, we take a hypothesis testing approach to understand the risk of inadvertently revealing the sensitive attribute by sharing log files. We design a set of *indistinguishability tests* that provide a user with an empirical understanding on the risk of leaking the sensitive attribute by sharing the log files. The tests do not modify the log files. Based on the test

results, a user can decide whether it is necessary to redact or obfuscate any metrics before releasing the logs.

This testing-based approach is similar to running a randomness test for a Pseudo-Random Number Generator (PRNG). The randomness test consists of a set of hypothesis tests that identify non-random patterns in psuedo-random sequences. If the randomness test fails, there is likely to be flaw in the design or implementation of the PRNG. Meanwhile, passing the randomness test only gives a user higher confidence in the quality of a PRNG. There is no guarantee for randomness even if the sequences pass all randomness tests. Similarly, passing the indistinguishability tests do not theoretically guarantee γ-log indistinguishability. However, it gives the user a higher confidence (quantified by γ) that the sensitive attribute is unlikely to be leaked through the released log files.

6.2.2 Steps of Indistinguishability Testing

The goal of indistinguishability testing is to identify violations of γ-log indistinguishability. Therefore, the null hypothesis ($\mathcal{H}_0$) under test is that P is γ-log indistinguishable on a candidate set C. Associated with this null hypothesis is the alternative hypothesis ($\mathcal{H}_\mathrm{A}$) that P is not γ-log indistinguishble. The testing process consists of three steps: (i) generation of test logs, (ii) selection of mapping functions, and (iii) performing hypothesis tests.

Step 1: Generate Test Logs. The first step is to obtain a set of logs to perform the tests on. For each sensitive attribute $x_i \in C$, a set of n parsed log files $\mathcal{L}_i = \left\{ l_i^{(k)} \;\middle|\; k \in \mathbb{Z}^+, k \leq N \right\}$ are generated by running the process for N times.

Step 2: Select Mapping Functions. Due to the high dimensionality of each parsed log file l, directly performing hypothesis tests on l requires an impractically large number of samples. Therefore, a *mapping function* $f : l \mapsto u$ is used to map l to a lower-dimensional vector u on which hypothesis tests can be efficiently performed. For example, the mapping function f_{length} returns the length of a log file (i.e. number of measurements over time). To improve test coverage, one should select a variety of mapping functions that cover different aspects of the log.

Step 3: Perform Hypothesis Tests. Finally, for each pair of $x_{i_1}, x_{i_2} \in C$, the user obtains two groups of outputs:

$$U_{i_1} = \left\{ f\left(l_{i_1}^{(k)}\right) \;\middle|\; l_{i_1}^{(k)} \in \mathcal{L}_{i_1} \right\}, \quad U_{i_2} = \left\{ f\left(l_{i_2}^{(k)}\right) \;\middle|\; l_{i_2}^{(k)} \in \mathcal{L}_{i_2} \right\}. \tag{6.1}$$

A hypothesis test T is performed on U_{i_1} and U_{i_2} to determine whether the output of the mapping function f is sufficient to distinguish between x_{i_1} and x_{i_2}.

Each indistinguishability test is a combination of a mapping function f and a hypothesis test T. In the following subsections, we propose a test suite consisting of different pairs of (f, T) that cover various aspects of a log file. The test suite can be used similarly to the randomness test suite [121]. Each test returns an independent result on whether there is a risk of information leakage. In Section 6.2.4, we propose an analytical approach that combines the test results and leads to a conclusion on the risk of revealing the sensitive attribute.

6.2.3 Designing Indistinguishability Tests

An indistinguishability test consists of two parts: a mapping function f that extracts information from an parsed log file l and a hypothesis test T that checks whether the extracted information leaks the sensitive attribute. In the following subsections, we propose four different types of mapping functions f and associate them with different hypothesis tests T. Specifically, when $\gamma = 0$, we perform kernel tests [122] and χ^2 tests [123] on the outputs of f. When $\gamma > 0$, we connect γ-indistinguishability with ε-differential privacy and perform differential privacy tests [124].

Mapping Functions

There are infinite number of functions that could extract useful information from a parsed log file, and it is impractical to design a "complete" set of mapping functions to cover all possible information leakage. Therefore, to strike a balance between the efficiency and completeness, we propose a set of mapping functions $\mathcal{F}$ that meets two criteria: (i) each mapping function $f \in \mathcal{F}$ returns a low-dimensional numerical/categorical vector; (ii) different mapping functions cover different aspects of l. The first criterion ensures that hypothesis tests can be efficiently performed on the output of f, while the second criterion reduces test redundancy and improves test completeness. This idea is similar to performing a set of randomness tests, where each test checks a different statistical pattern in a random sequence.

Length. The length of a parsed log file l refers to the number of measurements that have been recorded in the log. To extract this information for testing, we define a length mapping function f_{length} that returns the length of an input log file l.

Frequency (Categorical) Since there are few hypothesis tests that support comparison between multi-dimensional categorical vectors, we independently compare each categorical metric in l. A *frequency mapping function* $f_{\text{freq},j,t,w}$ returns the count of each value for the j-th categorical metric in the time window $[t, t+w)$. We use a set of frequency mapping functions to extract the information from a sequence of non-overlapping time windows: $\mathcal{F}_{\text{freq},j,w} = \{f_{\text{freq},j,t,w} \mid t = kw, k \in \mathbb{N}, k < L/w\}$, where L is the length of the log file. The output of each mapping function $f \in \mathcal{F}_{\text{freq},j,w}$ is a frequency vector containing the count for each value of the categorical metric. The window size w can be adjusted to balance between testing time and testing strength. A smaller w allows the user to identify minor differences between two sets of logs, but requires more tests to be performed. When $w = 1$, we perform one test on each measurement in l.

Moving Average (Numerical). When analyzing numerical metrics, we combine them into a multi-dimensional time series, and apply time series analysis techniques. The *moving average* technique replaces each element in a time series with the average of surrounding elements to eliminate local variations. Similarly, given a series of numerical metrics, we propose a *moving average mapping function* $f_{\text{avg},t,w}$ that calculates each metric's average value in the time window $[t, t+w)$. We use a set of moving average mapping functions to extract the information from a sequence of non-overlapping time windows: $\mathcal{F}_{\text{avg},w} = \{f_{\text{avg},t,w} \mid t = kw, k \in \mathbb{N}, k < L/w\}$. The output of each mapping function $f \in \mathcal{F}_{\text{avg},w}$ is a multi-dimensional numerical vector whose dimension equals to the number of numerical metrics in the log.

Moving Difference (Numerical). Local variations in a time series could also leak sensitive information. A common technique to study the local variation is to calculate the difference between consecutive measurements. Therefore, we propose a *moving difference mapping function* $f_{\text{diff},t}$ that returns the difference between a measurement at time $t+1$ and a measurement at time t for all the numerical metrics. We use a set of moving difference mapping functions to extract the difference between each consecutive measurements: $\mathcal{F}_{\text{diff}} = \{f_{\text{diff},t} \mid t \in \mathbb{N}, t < L-1\}$. Similar to the moving average function, the output of each mapping function $f \in \mathcal{F}_{\text{diff}}$ is a multi-dimensional numerical vector whose dimension equals to the number of numerical metrics in the log. When L is large, performing tests on the output of each function in $\mathcal{F}_{\text{diff},j}$ can be time-consuming. Therefore, we pick s mapping functions from $\mathcal{F}_{\text{diff}}$ uniformly at random, and obtain $\mathcal{F}_{\text{diff},s} \subseteq \mathcal{F}_{\text{diff}}$.

Considerations for Choosing Hypothesis Tests

For each mapping function f, we obtain two sets of outputs U_{i_1} and U_{i_2} (E.q. 6.1) that are associated with sensitive attributes x_{i_1} and x_{i_2} respectively. The next step is to select a two-sample hypothesis test T that takes U_{i_1}, U_{i_2} as inputs and identifies violations of γ-log indistinguishability. The hypothesis test T should meet two criteria:

1. *Correctness:* If the log-generating process P is γ-log indistinguishable, the probability of rejecting the null hypothesis under significance level α should be no greater than α.

2. *Power:* Among all known tests that satisfy the correctness criterion, the test with the strongest power should be selected.

The correctness criterion minimizes the type I error—the probability of rejecting log files that satisfy γ-indistinguishability. This criterion guarantees usability of the test. In software testing, if a unit test often fails on correct code, its result will be ignored by developers. Similarly, if a hypothesis test has a large type I error, the test result becomes unaccountable. To guarantee the correctness criterion, the null hypothesis of T ($\mathcal{H}_0$) needs to be a necessary condition for γ-log indistinguishability (i.e. $\mathcal{H}_0$ should always hold when P satisfies γ-log indistinguishability).

The power criterion minimizes the probability of accepting log files that violate γ-indistinguishability. If the process P passes a test with stronger power, the sensitive attribute is less likely to be leaked.

Tests of 0-Log Indistinguishability

When P is 0-log indistinguishable on $\{x_{i_1}, x_{i_2}\}$, the outputs U_{i_1}, U_{i_2} of any mapping function should have the same distribution. Otherwise, the difference between U_{i_1} and U_{i_2} would not allow the adversary to gain advantage in distinguishing between x_{i_1} and x_{i_2}. Therefore, given two samples U_{i_1}, U_{i_2}, the null hypothesis under test is

$$\mathcal{H}_0 \quad : \quad U_{i_1}, U_{i_2} \text{ have the same distribution.}$$

There are several hypothesis tests that can check whether two samples have the same distribution, such as the t-test [125], the KS test [126], χ^2 test [123], and kernel two-sample test [122]. Some of these tests, such as t-test, assume that the two samples come from a known distribution (e.g. normal distribution). Since these assumptions do not always hold in practice [120], we choose among non-parametric tests that do not rely on any assumptions

$\mathcal{F}$	**Size of $\mathcal{F}$**	**Output Format**	**Test**
Length	1	Integer	Kernel Test
Frequency	$\lceil L/w \rceil$	Count	χ^2 Test
Moving Average	$\lceil L/w \rceil$	Numerical vector	Kernel Test
Moving Difference	s	Numerical vector	Kernel Test

Table 6.1: Mapping Functions in Indistinguishability Tests.

about the underlying distribution of the samples. Specifically, we use χ^2 tests for categorical metrics and kernel two-sample tests for numerical metrics because they are shown to be more powerful than other tests that serve the same purpose [122]. Additionally, since kernel two-sample tests support comparisons between multi-dimensional numerical vectors, the outputs from multiple numerical metrics can be tested together. This approach could identify potential information leakage from the correlation between different metrics.

Table 6.1 shows the association between different mapping functions and the hypothesis tests to be performed on the outputs of these functions. Based on the types of mapping functions, we name the indistinguishability tests `length test`, `frequency test`, `moving average test`, and `moving difference test`.

DP Test

If the user is willing to tolerate a small amount of information loss when releasing the log, he could adjust the risk of revealing the sensitive attribute by setting the constant γ. However, when $\gamma > 0$, the tests in Section 6.2.3 are no longer applicable because U_{i_1} and U_{i_2} could come from slightly different distributions. Hence, we use DP tests [124] to check γ-log indistinguishability with $\gamma > 0$. Consider a mechanism $\mathcal{M}_{P,f} : x \mapsto u$ consisting of two steps: (i) generate a parsed log file l by running P with sensitive attribute x; (ii) compute the output $u = f(l)$. The following theorem shows the connection between γ-indistinguishability and DP.

Theorem 6.1. *If P is γ-indistinguishable on C, for all mapping function f, $\mathcal{M}_{P,f}$ is ε-differentially private on any pair of $x_{i_1}, x_{i_2} \in C$, where $\varepsilon = \log\left((1+\gamma)/(1-\gamma)\right)$.*

Therefore, the indistinguishability test fails if the DP test fails on any pair of U_{i_1}, U_{i_2} generated by the mapping functions in Section 6.2.3. DP tests are applicable to both categorical and numerical data and can be performed on the outputs of all mapping functions.

However, since the DP test is an adapted form of binomial tests [124], it is not as powerful as the tests listed in Section 6.2.3. Consequently, there is a higher risk that the tests could

incorrectly accept log files that are not γ-indistinguishable. Adapting more powerful statistical tests for DP and γ-log indistinguishability is nontrivial and retained for future work.

6.2.4 Interpretation of Test Results

Similar to a randomness test suite [121], an indistinguishability test suite consists of multiple tests that cover different aspects of a log file. Each test has a unique mapping function f that extracts some particular information from the log. Specifically, the mapping functions either (i) belong to different mapping function sets $\mathcal{F}$ (Table 6.1); or (ii) extract information from different sub-sequences in l (e.g., mapping functions in $\mathcal{F}_{\text{avg},w}$ calculate the moving average in different time windows). Based on these two differences, we introduce two methods to combine multiple test results.

Combining Test Results over Different Sub-Sequences Suppose $(p_1, p_2, \ldots, p_k)$ is a sequence of p-values returned by performing the same type of indistinguishability test on different sub-sequences of the logs. Under the null hypothesis (i.e., the log-generating process is γ-log indistinguishable), the p-values should be uniformly distributed [127]. We use the Fisher's method [128] to test the uniformity of the p-values.

Combining Test Results over Different Mapping Function Sets. By combining the results of tests performed on different sub-sequences, we obtain one p-value associated with each mapping function set. Since different mapping function sets focus on different metrics/statistics in the logs, each set represents a unique attack surface for the adversary. For example, if the test associated with the length mapping function fails, it is likely that the adversary could infer the sensitive attribute based on the length of the log. Therefore, the p-value associated with different mapping function sets should be interpreted independently. If any of the p-value is smaller than the significance level α, there is a risk of revealing the sensitive attribute. Additionally, failed tests also indicate the statistics associated with the information leakage. In Section 6.3, we introduce obfuscation methods to mitigate the risk identified by each test.

6.3 PROTECTIONS WITH LOG OBFUSCATION

In practice, companies often apply obfuscation techniques to hide sensitive information in log. For example, Reiss et al. [9] proposed log-obfuscation techniques for releasing the Google's cluster traces. However, their approach is limited by the lack of understanding on

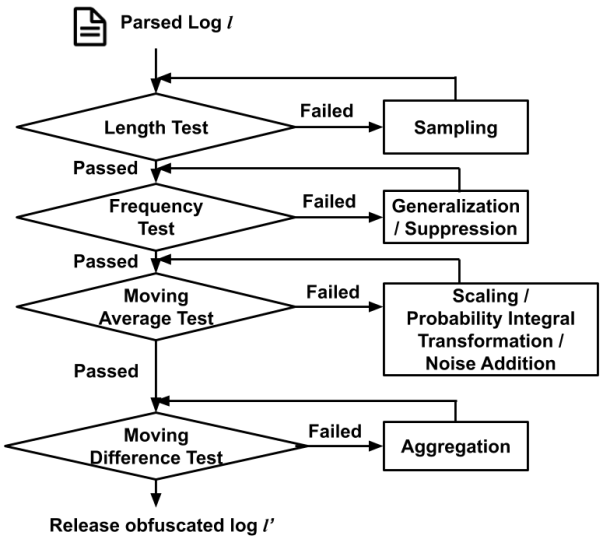


Figure 6.2: Testing-Based Log Obfuscation Framework

the protection goal of each obfuscation technique. Therefore, it is unclear whether these techniques are effective in hiding the sensitive information and whether it is necessary to apply them under a different system or a different protection goal.

In this section, we combine the indistinguishability tests proposed in Section 6.2.3 with different obfuscation techniques. The results of the tests suggest whether a specific obfuscation technique would be helpful in protecting the user-specified sensitive attributes. Additionally, we find out that existing obfuscation techniques are not sufficient to hide all the information identified by our indistinguishability tests and propose two novel obfuscation techniques: probability integral transformation and noise addition. By combining these techniques, we propose an end-to-end framework (Figure 6.2) to help users identify sources of potential information leakage and mitigate it. Specifically, we use sampling to mitigate leakage identified by length tests, generalization and suppression to mitigate leakage identified by frequency tests, and aggregation to mitigate leakage identified by moving difference tests. Meanwhile, a moving average test could identify leakage from three possible sources: (i) the magnitude of a numerical metric; (ii) the distribution (e.g. variation and skewness) of a numerical metric; (iii) the correlations between different numerical metrics. Therefore, we use three different obfuscation methods to mitigate the leakage associated with each source: (i) sampling, (ii) probability integral transformation; (iii) noise addition.

Sampling. Sampling refers to the process of selecting a subset of measurements in a log file. For example, the user could release a set of measurements uniformly sampled from the whole log file.

Generalization and Suppression. Generalization and suppression are often used to hide sensitive information in categorical attributes [5]. They can be applied to categorical metrics when information leakage is identified by a frequency test.

Scaling. Scaling could mitigate the information leakage from the magnitude of a numerical metric. For example, in the released Google cluster trace dataset, Reiss et al. [9] re-scaled the machine capacity data to guarantee that the maximum observed value is 1.

Probability Integral Transformation. Suppose X is a random variable with cumulative distribution function (cdf) F_X. Probability integral transformation (PIT) converts X to a different random variable Y with cdf F_Y. It relies on the property that $Z = F_X(X)$ follows a uniform distribution. Therefore, the random variable $Y = F_Y^{-1}(F_X(X))$ follows the distribution defined by F_Y.

Suppose S_1, S_2, S_3 are three sets of measurements obtained under sensitive attributes x_1, x_2, x_3 respectively. To perform the obfuscation, one needs to first estimates the empirical cdf F_{mix} of $S_1 \cup S_2 \cup S_3$. Then, for each set S_i ($i \in \{1, 2, 3\}$), by applying PIT on each value in S_i, one obtains obfuscated measurements that follow the new distribution defined by cdf F_{mix}. PIT mitigates the information leakage from the distribution of a numerical metric by ensuring that the measurement taken under different sensitive attributes share the same distribution.

Noise Addition. The key limitation of PIT is that it needs to be *independently* applied to each numerical metrics. Hence, PIT could not prevent information leakage from the correlation between different attributes. For example, suppose two numerical metrics m_1 and m_2 have a positive correlation. This correlation would be retained after PIT. Therefore, if this correlation leaks the sensitive attribute, one needs to add noise to both m_1 and m_2 to hide it. We add Gaussian noise that follows the distribution $\mathcal{N}(0, \sigma^2)$, and determined σ based on the result of the moving average test.

Aggregation. Failing of the moving difference test indicates that the local variations in the numerical metrics might reveal the sensitive attribute. Hence, one could use aggregation techniques to prevent the leakage. For example, instead of revealing all the measurements, one could calculate the average of a measurement over a fixed-length time window. The aggregation technique eliminates local variations, but preserves the trend of the metrics.

6.4 CASE STUDIES

In this section, we performed case studies on two log datasets: a Spark [117] event log dataset and a hardware performance counter (HPC) dataset collected by Ganju et al. [129]. Similar

Tests	**Length**		**Frequency**		**Moving Average**		**Moving Diff.**	
γ	0	0.2	0	0.2	0	0.2	0	0.2
p-values	0.81	0.75	0.89	1.00	0.64	1.00	0.62	1.00
Testing Time (s)	<0.01	0.22	0.03	13.05	5.14	52.32	4.74	66.60

Table 6.2: Test Correctness and Performance Analysis on Spark Event Logs.

Tests	**Length Test**			**Frequency Test**		
γ	0	0.2	0.4	0	0.2	0.4
No Obfuscation	<0.01	<0.01	<0.01	-	-	-
Sampling	0.99	0.77	0.90	0.93	1.00	1.00
Sampling + Scaling	0.99	0.78	**0.91**	0.93	1.00	**1.00**
Sampling + Scaling + PIT	**0.99**	**0.79**	0.93	**0.93**	**1.00**	1.00
Tests	**Moving Average Test**			**Moving Difference Test**		
γ	0	0.2	0.4	0	0.2	0.4
Sampling	<0.01	<0.01	<0.01	-	-	-
Sampling + Scaling	<0.01	<0.01	**0.17**	-	-	**0.69**
Sampling + Scaling + PIT	**0.27**	**1.00**	1.00	**1.00**	**1.00**	1.00

Table 6.3: Effectiveness of Different Obfuscation Techniques on Spark Event Logs.

to Google cluster traces [9, 130], Spark event logs provides a variety of performance counters and workload traces. These traces are extensively used to perform performance analysis and modeling [131, 132] for different purposes such as performance prediction, diagnosis and configuration selection[133, 134, 135]. The HPC dataset contains measurements about hardware events on a computer system. These measurements have been demonstrated to be useful in detecting malware [136], side-channel attacks [137], and cryptomining behavior [138]. These two case studies cover different data types and different correlations between the sensitive information and the logged metrics.

We evaluated our framework from four aspects: the correctness of the tests, the effectiveness of obfuscation, the risk under attacks, and the utility of the obfuscated logs. Specifically, our evaluations on the obfuscation techniques focused on the two novel techniques proposed in this chapter: probability integral transformation (PIT) and noise addition. We also used sampling and scaling to mitigate the information leakage. Our case studies did not cover generalization, suppression, and aggregation.

6.4.1 Spark Event Log Dataset

Experimental Setup Spark is an open-source distributed system for large data analysis. A Spark application is automatically divided into several stages, each consisting of multiple tasks that can be executed in parallel. A Spark event log records information about each individual task in a Spark application. We adapted Spark trace analysis tool [131] to parse the event logs into structured multidimensional sequential vectors, where each data point represents a different task. Each parsed log file contains 15 numerical metrics and 4 categorical metrics [1]. The log-generating process P was a Spark application for training a multi-layer perceptron (MLP) model using the SparkML library. The training data were randomly generated using the `sklearn` library. The training dataset contained 10,000 records with 100 features and 2 classes. We considered hardware information as the sensitive attribute. Specifically, we ran the same process P on two clusters of machines provided by Emulab (`emulab.net`):

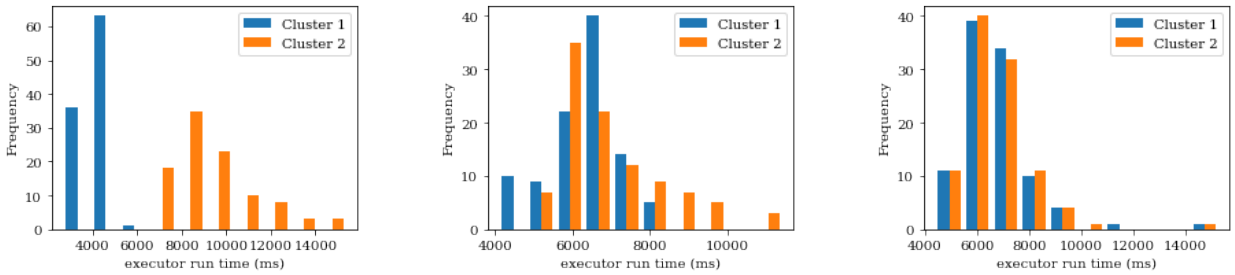
- **Cluster 1:** 10 Dell Poweredge R430 1U servers each with two 2.4 GHz 64-bit 8-Core processors and 64GB RAM;
- **Cluster 2:** 10 Dell PowerEdge 2850s each with a single 3GHz processor and 2GB RAM.

The protection goal was to prevent the adversary from correctly guessing the cluster on which P was running.

On each cluster, we obtained 100 log files by repeatedly running P under the same settings. $\mathcal{L}_1$ denotes the set of logs obtained on Cluster 1, and $\mathcal{L}_2$ denotes the set of logs obtained on Cluster 2. Since parallel tasks of the same job often share similar system metrics (e.g. running time, I/O size), we randomly sampled 1 task per stage when performing the moving average tests and the moving difference tests to prevent test redundancy. The sampled time sequences have a maximum length of 48. We set the window size $w = 1$ for frequency tests and moving average tests. All the tests were performed locally on a laptop with single 2.7 GHz Intel Core i5 processor.

Correctness and Performance We performed experiments to demonstrate that the indistinguishability tests have low testing overhead, and the tests could stably accept the null hypothesis $\mathcal{H}_0$ (i.e., P is γ-log indistingsuihable) when $\mathcal{H}_0$ holds. An indistingusihability test is correct only if it rejects the null hypothesis with a low probability ($\Pr[p < \alpha \mid \mathcal{H}_0] < \alpha$) when P satisfies γ-log indistinguishability. To check the correctness of the tests, we performed the tests on two groups of logs generated on the same cluster. Specifically, we randomly

[1] A complete metric list is available on `spark.apache.org/docs/latest/monitoring.html`



(a) Original (b) Scaled (c) Scaled + PIT

Figure 6.3: Analysis on Probability Integral Transformation (PIT).

divided $\mathcal{L}_1$ into two groups G_1 and G_2, each containing 50 parsed log files. We performed indistinguishability tests on G_1 and G_2 with the null hypothesis that G_1 and G_2 are γ-log indistinguishable. We repeated the testing process for 10 times with $\alpha = 0.01$. In all the 10 repetitions, the log files passed the indistinguishability tests for both $\gamma = 0$ and $\gamma = 0.2$. Table 6.2 presents the average p-values for each test.

Additionally, we evaluated the average testing time for each test (Table 6.2). The total test time is around 10 seconds when $\gamma = 0$ and around 2 minutes when $\gamma > 0$. When $\gamma = 0$, the tests took less time because χ^2 tests and kernel tests are more efficient than DP tests on multi-dimensional data.

Testing-Based Obfuscation We performed different obfuscation techniques based on the test results and demonstrated that these techniques could effectively reduce the risk of information leakage. We evaluated the log-sharing framework on the two sets of logs $\mathcal{L}_1$ and $\mathcal{L}_2$ obtained on different clusters. We followed the iterative process shown in Figure 6.2: when the logs failed an indistinguishability test, we applied obfuscation techniques and repeated the failed tests; when the logs passed a test, we moved to the next test. The logs could be shared after all tests are passed.

We applied three different obfuscation techniques. When the length test failed, we sampled 3 parallel tasks per stage and discarded measurements of remaining tasks in the log file. When the moving average test failed, we scaled the numerical metrics based on the following strategy: (i) we calculated the medians m_1 and m_2 in $\mathcal{L}_1$ and $\mathcal{L}_2$ for each numerical metric; (ii) we scaled each numerical metric in log $\mathcal{L}_i$ by multiplying it with a constant $c_i = \frac{m_1+m_2}{2m_i}$. This scaling strategy ensures that numerical metrics in $\mathcal{L}_1$ and $\mathcal{L}_2$ share the same median ($\frac{m_1+m_2}{2}$) while preserving the relative magnitudes of these metrics. If the scaled logs still failed the moving average tests, we further applied PIT to ensure that the metrics in $\mathcal{L}_1$ and $\mathcal{L}_2$ share the same distribution (Section 6.3).

We performed indistinguishability tests with $\alpha = 0.01$ under three different settings: (i) $\gamma = 0$, (ii) $\gamma = 0.2$, and (iii) $\gamma = 0.4$. Based on the definition of γ-log indistinguishability, the settings could be translated into three levels of protection objectives against an adversary trying to infer the sensitive attribute: (i) the adversary's performance should be equivalent to random guessing; (ii) the attack accuracy should be lower than 0.6; (iii) the attack accuracy should be lower than 0.8.

Table 6.3 presents the p-value of each test under different γ and obfuscation techniques. The values in bold indicate that the obfuscated logs passed all tests (i.e., $p > 0.01$) and could be shared. When $\gamma = 0$ and $\gamma = 0.2$, sampling, scaling, and PIT were required to achieve the protection criteria. Meanwhile, when $\gamma = 0.6$, PIT was not needed to ensure the protection criterion.

When no obfuscation was applied, logs generated on Cluster 1 and Cluster 2 had different length. Since machines in Cluster 1 have more processors compared to machines in Cluster 2, they could support more parallel tasks, which resulted in a larger log file. This leakage was mitigated by sampling a subset of tasks per stage.

However, after sampling, the logs still leaked sensitive information through the magnitude of numerical metrics. Since the same job was divided into more parallel tasks on Cluster 1, tasks running on Cluster 1 had shorter duration and smaller I/O size. An adversary could use this information to infer the cluster on which the logs were generated. Scaling was not sufficient to prevent the leakage because the task metrics on Cluster 1 and Cluster 2 were different not only in their magnitudes but also in the variation and skewness of their distributions.

Figure 6.3 shows the distributions of `executor run time (ms)` of Task 0 in Stage 0. Prior to the obfuscation (Figure 6.3a), the distributions were different in two aspects: (i) tasks on Cluster 1 had shorter runtime than tasks on Cluster 2; (ii) runtime on Cluster 2 had a right-skewed distribution, indicating that there were more stragglers (i.e., tasks with runtime larger than 1.5× the median runtime). The scaled metrics (Figure 6.3b) shared similar magnitudes, but did not eliminate the stragglers on Cluster 2. Therefore, an adversary could infer the sensitive information by identifying the stragglers in the log file. After applying PIT (Figure 6.3c), the number of stragglers increased on Cluster 1 and decreased on Cluster 2. Since the two sets of obfuscated logs shared the same distribution, an adversary could no longer infer the cluster on which the logs were generated.

Risk under Different Attacks We designed three different attacks to verify the test results. We showed that an adversary could correctly infer the sensitive attribute when the tests failed. Moreover, when the logs were obfuscated and passed the tests, the attack

	Attack Acc.			# of Correct Ans.		
	MLP	NN	Length	<10	10-15	16
No Obfuscation	1.00	1.00	1.00	0	0	100%
Sampling	0.95	1.00	0.55	0	27%	73%
Sampling + Scaling	0.58	0.65	0.55	0	27.5%	72.5%
Sampling + Scaling + PIT	0.43	0.55	0.55	0.5%	30.5%	69%

Table 6.4: Attack and Utility Analysis on Spark Logs.

accuracy dropped to around 0.5.

Specifically, we modeled an attack as a classification problem with the parsed log files as input features and the cluster information as class labels. We randomly divided $\mathcal{L}_1 \cup \mathcal{L}_2$ into equal-sized datasets $\mathcal{L}_{train}$ and $\mathcal{L}_{test}$ each containing 50 parsed logs from Cluster 1 and 50 parsed logs from Cluster 2. We assumed that an adversary had access to the labels (i.e. cluster information) for logs in $\mathcal{L}_{train}$ and wanted to predict the labels of logs in $\mathcal{L}_{test}$. In practice, an adversary could obtain $\mathcal{L}_{train}$ and their labels by running P on her own machines, and $\mathcal{L}_{test}$ represents the logs shared by the users.

We trained three different attack classifiers: (i) a multi-layer perceptron (MLP) model with 100 hidden units; (ii) a nearest neighbor (NN) classifier; and (iii) a length classifier that predicts the cluster label based on the length of a parsed log file. Table 6.4 shows the attack accuracy on $\mathcal{L}_{test}$ of the three attacks. Without obfuscation, all attacks achieved high accuracy. Sampling could only protect against the length attack, and scaling could reduce the attack accuracy to below 0.7. This result complies with our previous test results for $\gamma = 0.4$. After PIT, all attack accuracy dropped to around 0.5, indicating that the obfuscated logs were likely to satisfy 0-indistinguishability.

Utility of Obfuscated Logs We demonstrated that the obfuscated logs were still useful in helping users identify bottlenecks of the system. We performed the analysis proposed by Ousterhout et al [131] to study the utility of obfuscated logs. The goal of the analysis is to identify performance bottlenecks in a system. It estimates how long the application is blocked on a certain aspect of the system (e.g. network, computation, I/O). To get a quantitative understanding of the analysis results, we designed 16 questions that could be answered based on the analysis results:

(1) What is the bottleneck of the system?

(2-4) Is blocked time on computation greater than 20%/50%/80%?

(5-7) Is blocked time on disk greater than 20%/50%/80%?

(8-10) Is blocked time on GC greater than 20%/50%/80%?

(11-13) Is blocked time on network greater than 20%/50%/80%?

(13-15) Is blocked time on stragglers (i.e., tasks with runtime larger than 1.5× the median runtime) greater than 20%/50%/80%?

(16) What is the number of stragglers?

We performed the analysis for each $l \in \mathcal{L}_1 \cup \mathcal{L}_2$, and compared the answers obtained from original logs and obfuscated logs. Table 6.4 presents the number of questions that could be correctly answered on the obfuscated logs. After obfuscation, the analysis process could still correctly answer all the questions on around 70% of the obfuscated log files. Among the 16 questions, the questions about stragglers (Q13-Q16) were most likely to be influenced by the obfuscation techniques. Since tasks on Cluster 2 had more stragglers compared to tasks on Cluster 1, giving out accurate information about the stragglers would inevitably leak information about the clusters. Therefore, there is a trade-off between hiding the sensitive information and retaining useful information in obfuscated logs. Our testing-based framework provides a better understanding on this trade-off by helping users intuitively understand the effectiveness of different obfuscation techniques.

6.4.2 Hardware Performance Counter Dataset

Experimental Setup In this case study, we studied the risk associated with sharing HPC datasets. For example, suppose a small company wants to detect whether its employees have been using the company resources for covert cryptomining. The company could outsource this detection to a third-party service by sharing its HPCs. However, the company might be concerned about leaking information about the security vulnerabilities of its machines. In this case, the sensitive attribute to be protected is whether the machines are patched against Spectre [139] and Meltdown [140] attacks.

Our dataset [129] includes 22 different hardware counters (i.e., numerical metrics) for a variety of cryptomining and non-mining applications. Each record in the dataset contains 3-5 measurements taken at an interval of 2 seconds. The dataset contains two parts: the unpatched subset was generated by running applications on a machine vulnerable to Spectre and Meltdown attacks; the patched subset was generated by running the same applications on the same machine after the patches have been installed. We generated $\mathcal{L}_{\text{p}}$ and $\mathcal{L}_{\text{up}}$

Tests	**Length**		**Moving Average**		**Moving Difference**	
γ	0	0.2	0	0.2	0	0.2
p**-values**	1.00	1.00	0.71	0.99	0.67	0.98
Time (s)	<0.01	0.04	0.66	15.01	0.53	12.31

Table 6.5: Correctness and Performance on HPCs.

Tests	**Length**		**Moving Average**		**Moving Difference**	
γ	0	0.2	0	0.2	0	0.2
No Obfuscation	0.89	1.00	<0.01	<0.01	-	-
Scaling	0.89	1.00	<0.01	<0.01	-	-
PIT	0.89	**1.00**	<0.01	**0.99**	-	**0.72**
Scaling + PIT	0.89	1.00	<0.01	0.99	-	0.81
PIT + Noise	**0.89**	1.00	**0.02**	0.92	**0.48**	0.86

Table 6.6: Effectiveness of Different Obfuscation on HPCs

by randomly sampling from the patched and unpatched subsets respectively. $\mathcal{L}_{\text{p}}$ and $\mathcal{L}_{\text{up}}$ both contain 50 records from cryptomining applications and 50 records from non-mining applications. The sampled time sequences have a maximum length of 48.

Correctness and Performance We demonstrated that the tests were correct and they incurred little testing overhead on the HPC dataset. We randomly divided $\mathcal{L}_{\text{p}}$ into two groups G_1 and G_2. We performed indistinguishability tests on G_1 and G_2 with the null hypothesis that they are γ-log indistinguishable with $\gamma = 0$ and $\gamma = 0.2$. We repeated the testing process for 10 times with $\alpha = 0.01$. In all the 10 repetitions, the indistinguishability tests were passed for both $\gamma = 0$ and $\gamma = 0.2$. Table 6.5 presents the average p-values and running time for each test.

Testing-Based Obfuscation We showed that the obfuscation techniques that were effective on the Spark event logs could not protect the HPC dataset, but the sensitive information could be effectively hidden by the noise addition technique we proposed.

We applied three obfuscation techniques to protect the HPC dataset: scaling, PIT, and noise addition. Table 6.6 shows the testing results after each obfuscation technique was applied. We performed indistinguishability tests with $\alpha = 0.01$. The values in bold indicate that the obfuscated logs passed all tests and could be shared. Unlike the Spark event log dataset, the HPC dataset could not be protected by scaling because the main source of

	MLP Attack	NN Attack	Utility
No Obfuscation	0.60	0.80	0.96
Scaling	0.95	0.93	0.97
Scaling + PIT	0.60	0.58	0.91
PIT + Noise	0.47	0.57	0.90

Table 6.7: Attack and Utility Analysis on Obfuscated HPCs.

information leakage is the distribution, rather than magnitude of the performance counters. For example, compared to unpatched machines, patched machines have a larger variation in the number of executed instructions per second. Moreover, the sensitive attribute in the HPC dataset could be inferred through the correlations between different performance counters, so PIT is not sufficient to protect the information leakage.

To hide the correlations that could leak the sensitive attribute, we added Gaussian noise $\mathcal{N}(0, \sigma^2)$ to each measurements in the HPC dataset. To ensure that the noise we added have relatively same magnitude as the original value, we normalized the metrics prior to noise addition and scaled them back after noise was added. We gradually increased σ until the moving average test was passed. With $\sigma = 0.09$, the moving average test was passed with $p = 0.02$. To reduce false negatives (i.e., incorrectly accept the null hypothesis), we repeated the testing process for 10 times and ensured that the test was passed in all the repetitions.

Risk and Utility of Obfuscated Logs We showed that the noise addition technique was effective in protecting against the attacks and incurred little utility loss on the HPC dataset.

Using the techniques in Section 6.3, we trained two attack classifiers to predict whether the machines were patched: (i) a multi-layer perceptron (MLP) model with 100 hidden units and (ii) a nearest neighbor (NN) classifier. We did not evaluate the length attack because all the records in the HPC datasets have similar length.

We evaluated the utility of the obfuscated dataset using an MLP model with 100 hidden units. The model was trained to predict whether a record was generated by a cryptomining or non-mining application. Table 6.7 presents the attack accuracy and utility under different obfuscation techniques. The attack accuracy conforms with the test results presented in Table 6.6, the obfuscation techniques reduced the maximum attack accuracy from 80% to 57% but still preserved a 90% accuracy in classifying mining and non-mining applications.

Analysis on Noise Addition We showed that PIT could effectively reduce the amount of noise required for protecting the sensitive information. Noise addition could mitigate any information leakage if the standard deviation of the noise (σ) is large enough. However,

	No Obfuscation	**Scaling**	**PIT**
$\sigma_{\min}$	0.86	0.89	**0.09**
Utility	0.62	0.54	**0.90**

Table 6.8: Obfuscation Prior to Noise Addition.

adding noise with large σ would incur huge utility loss on the logs. To minimize the utility loss, we applied other obfuscation techniques such as scaling and PIT prior to noise addition. Table 6.8 presents the effectiveness of these techniques in reducing the amount of required noise to pass the moving average test. By applying PIT prior to noise addition, we increased the utility accuracy by around 30%.

6.5 CONCLUSION

We have proposed a test-based framework to identify and mitigate the risk of information leakage in general-purpose log sharing. Our framework contains a set of statistical tests to identify violations of the log indistinguishability property and a variety of obfuscation methods such as probability integral transformation and noise addition. We tested our framework on Spark event logs and logs generated by a hardware performance counter. The framework effectively identified risks in information leakage and mitigated the risks with two obfuscation techniques.

CHAPTER 7: SCALABLE DP GENERATIVE MODEL VIA PATE

In this chapter, we present a novel approach G-PATE for training a scalable differentially private data generator, which can be used to produce synthetic datasets with strong privacy guarantee while preserving high data utility. Our approach leverages generative adversarial nets to generate data and exploits the PATE (Private Aggregation of Teacher Ensembles) framework to protect data privacy. Compared to existing methods, our approach significantly improves the use of privacy budget. This is possible since we only need to ensure differential privacy for the generator, which is the part of the model that actually needs to be published for private data generation. In particular, we connect a student generator with an ensemble of teacher discriminators and propose a private gradient aggregation mechanism to ensure differential privacy on all the information that flows from the teacher discriminators to the student generator. Theoretically, we prove that G-PATE ensures differential privacy for the data generator. Empirically, we provide thorough experiments to demonstrate the superiority of our method over prior work on both image and non-image datasets. This chapter is based on joint work with Boxin Wang, Zhuolin Yang, Kaizhao Liang, Shuang Yang, Bhavya Kailkhura, Carl Gunter, and Bo Li [141].

Machine learning has been applied to a wide range of applications such as face recognition [1], autonomous driving [2], and medical diagnoses [3, 4]. However, most learning methods rely on the availability of large-scale training datasets containing sensitive information such as personal photos or medical records. Therefore, such sensitive datasets are often hard to be shared due to privacy concerns. To handle this challenge, data providers sometimes release synthetic datasets produced by generative models learned on the original data. Though recent studies show that generative models such as generative adversarial networks (GAN) [142] can generate synthetic records that are indistinguishable from the original data distribution, there is no theoretical guarantee on the privacy protection. While privacy definitions such as differential privacy [143] and Rényi differential privacy [144] provide rigorous privacy guarantee, applying them to synthetic data generation is nontrivial.

Recently, two approaches have been proposed to combine differential privacy with synthetic data generation: DP-GAN [36] and PATE-GAN [37]. DP-GAN modifies GAN by training the discriminator using differentially private stochastic gradient descent. Though it achieves privacy guarantee due to the post processing property [38] of differential privacy, DP-GAN incurs significant utility loss on the synthetic data, especially when the privacy budget is low. In contrast, PATE-GAN trains differentially private GAN using the PATE mechanism [145].

Specifically, it first trains a set of teacher discriminators and then train a student discriminator based on the trained ensemble of teacher discriminators. To ensure differential privacy, the student discriminator is only trained on records that are produced by the generator and labeled by the teacher discriminators. The key limitation of this approach is that it relies on the assumption that the generator would be able to generate the entire "real" records space to bootstrap the training process. If most of the synthetic records are labeled as fake by the teacher discriminators, the student discriminator would be trained on a biased dataset and fail to learn the true data distribution. Consequently, this trained generator would not be able to produce high-quality synthetic data. This problem does not exist for traditional GAN, where the discriminator is always able to provide useful information to the generator since they can access the *real data records* rather than the synthetic data only. In addition, the two stage training process of PATE-GAN makes it less scalable or flexible in terms of varying the number of teacher discriminators.

The main contribution of this chapter is a new approach named G-PATE for training a differentially private data generator by combining the generative model with PATE mechanism. Our approach is based on the key observation that: *It is not necessary to ensure differential privacy for the discriminator in order to train a differentially private generator*. As long as we ensure differential privacy on the information flow from the discriminator to the generator, it is sufficient to guarantee the privacy property for the generator. To achieve this, we propose a private gradient aggregation mechanism to ensure differential privacy on all the information that flows from the teacher discriminators to the student generator. Compared to PATE-GAN, our approach has three advantages. First, it improves the use of privacy budget by only applying it to the part of the model that actually needs to be released for data generation. Second, our discriminator can be trained on original data records since it does not need to satisfy differential privacy. Finally, G-PATE is much more scalable given its simple architecture.

Theoretically, we show that our algorithm ensures differential privacy for the generator. Empirically, we conduct extensive experiments on the standard Kaggle credit card fraud detection dataset, as well as two image datasets MNIST and Fashion-MNIST. To the best of our knowledge, this is the first work that is able to scale to high-dimensional face image dataset such as CelebA [108] while still preserve high data utility. The results show that our method significantly outperforms all baselines including DP-GAN and PATE-GAN.

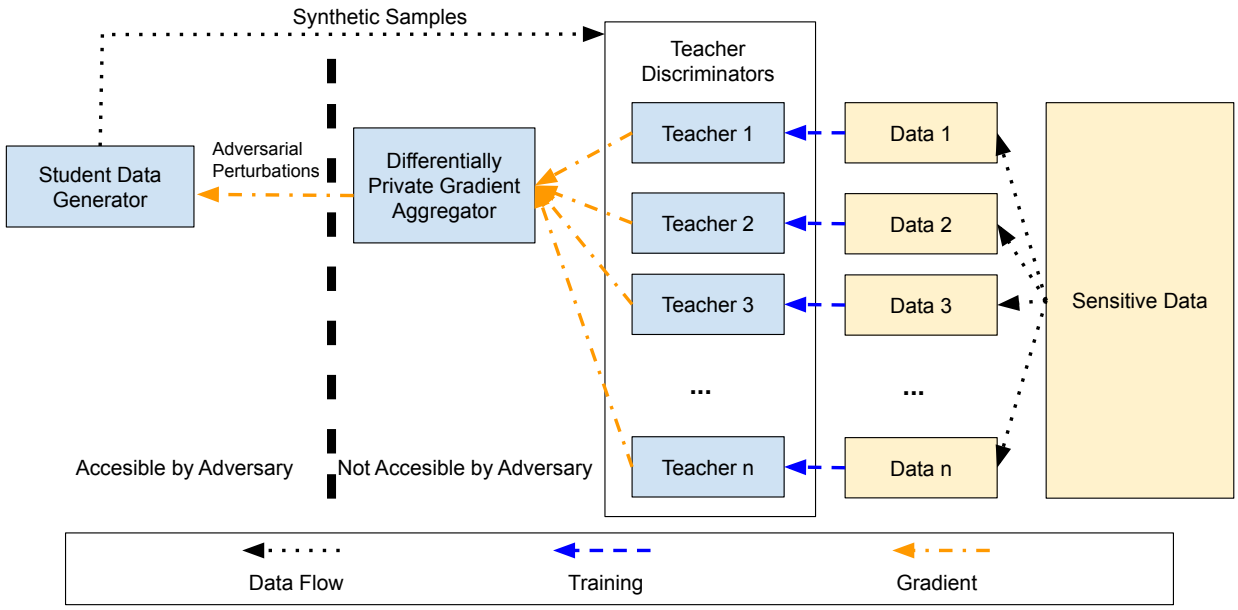


Figure 7.1: **Model Overview of G-PATE.** The model contains three parts: a student data generator, a differentially private gradient aggregator, and an ensemble of teacher discriminators.

7.1 THE G-PATE METHOD

In this section, we present our method named G-PATE. An overview of the method is shown in Figure 7.1. Unlike PATE-GAN and DP-GAN, G-PATE ensures differential privacy for the information flow from the discriminator to the generator. This improvement incurs less utility loss on the synthetic samples, so it can generate synthetic samples for higher dimensional and more complex datasets.

G-PATE makes two major modifications on the training process of GAN. First, we replace the discriminator in GAN with an ensemble of teacher discriminators trained on disjoint subsets of the sensitive data. The teacher discriminators do not need to be published, thus can be trained using non-private algorithms. In addition, we design a gradient aggregator to collect information from teacher discriminators and combine them in a differentially private fashion. The output of the aggregator is a gradient vector that guides the student generator to improve its synthetic samples.

Unlike PATE-GAN, G-PATE does not require any student discriminator. The teacher discriminators are directly connected to the student generator. The gradient aggregator sanitizes the information flow from the teacher discriminators to the student generator to ensure differential privacy. This way, G-PATE uses privacy budget more efficiently and better approximates the real data distribution to ensure high data utility.

7.1.1 Training the Student Generator

To achieve better privacy budget efficiency, G-PATE only ensures differential privacy for the generator and allows the discriminators to learn private information. The privacy property is achieved by sanitizing all information propagated from the discriminators to the generator. To ease privacy analysis, we decompose G-PATE into three parts: the teacher discriminators, the student generator, and the gradient aggregator. To prevent the propagation of private information, the student generator does not have direct access to any information in any of the teacher discriminators. Consequently, we cannot train the student generator by ascending its gradient based on loss of the discriminators. To solve this problem, we propose the use of *adversarial perturbation*, which is a small manipulation on the fake record x that causes the discriminator's loss on x to increase. The adversarial perturbation is calculated by ascending x's gradients on the loss of the discriminator. It teaches the student generator how to improve its fake records. In each training iteration, the student generator is updated in three steps: (1) A teacher discriminator generates adversarial perturbations for each record produced by the student generator. (2) The gradient aggregator takes the adversarial perturbations from all teacher models and generates a differentially private aggregation of them. (3) The student generator updates its weights based on the privately aggregated adversarial perturbation. The process is formally presented in Algorithm 7.1.

Generating Adversarial Perturbations. Let D be a teacher discriminator. Given a fake record x, we use $\mathcal{L}_D(x)$ to represent D's loss on x. In each training iteration, the weights of D are updated by descending their stochastic gradients on $\mathcal{L}_D$.

For each input fake record x, we generate an adversarial perturbation Δx that guides the student generator on improving its output. By applying the perturbation on its output, the student generator would get an improved fake record $\hat{x} = x + \Delta x$ on which D has a higher loss. Therefore, Δx is calculated as x's gradients on $\mathcal{L}_D$:

$$\Delta x = \left.\frac{\partial \mathcal{L}_D(a)}{\partial a}\right|_{a=x}. \tag{7.1}$$

With the adversarial perturbation Δx, the student generator can be trained without direct access to the discriminator's loss.

Updating the Student Generator. A student generator G learns to map a random input z to a fake record $x = G(z)$ so that x is indistinguishable from a real record by D. Given an adversarial perturbation Δx, the teacher discriminators have higher loss on the perturbed fake record $\hat{x} = x + \Delta x$ compared to the original fake record x. Therefore, the student generator learns to improve its fake records by minimizing the mean squared error

Algorithm 7.1 - Training the Student Generator.

Require: batch size m, number of teacher models n, gradient aggregator `Agg`, disjoint subsets of sensitive data $d_1, d_2, \ldots, d_n$

for number of training iterations **do**
  Sample m noise samples $\{z_1, z_2, \ldots, z_m\}$
  Generate fake samples $\{G(z_1), G(z_2), \ldots, G(z_m)\}$
  for $i \in \{1, \ldots, n\}$ **do**
    Sample m data samples $\{x_1, x_2, \ldots, x_m\}$ from d_i
    Update the teacher discriminator D_i by descending its stochastic gradient on $\mathcal{L}_{D_i}$ on both fake samples and real samples
    for $j \in \{1, \ldots, m\}$ **do**
      Calculate the adversarial perturbation $\Delta x_j^{(i)}$ as x_j's gradients on $\mathcal{L}_{D_i}(x_j)$
    end for
  end for
  for $j \in \{1, \ldots, m\}$ **do**
    $\Delta x_j \leftarrow \texttt{DPGradAgg}(\Delta x_j^{(1)}, \Delta x_j^{(2)}, \ldots, \Delta x_j^{(n)})$
    $\hat{x}_j \leftarrow G(z_j) + \Delta x_j$
  end for
  Update the student generator G by descending its stochastic gradient on $\mathcal{L}_G$ on $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m\}$
end for

(MSE) between its output $G(z)$ and the perturbed fake record $\hat{x}$.

$$\mathcal{L}_G(z, \hat{x}) = \frac{1}{k} \sum_{i=1}^{k} (G(z_i) - \hat{x}_i)^2, \tag{7.2}$$

where k is the number of synthetic records generated per training iteration. To ensure differential privacy, instead of receiving the adversarial perturbation from a single discriminator, we train the student generator using a differentially private gradient aggregator that combines adversarial perturbations from multiple teacher discriminators. Details are provided in Section 7.1.2.

7.1.2 Differentially Private Gradient Aggregation for G-PATE

G-PATE consists of a student generator and an ensemble of teacher discriminators trained on disjoint subsets of the sensitive data. In each training iteration, each teacher discriminator generates an adversarial perturbation Δx that guides the student generator on improving its output records. Different from traditional GAN, in G-PATE, the student generator does not have access to the loss of any teacher discriminators, and the adversarial perturbation is

the only information propagated from the teacher discriminators to the student generator. Therefore, to achieve differential privacy, it suffices to add noise during the aggregation of the adversarial perturbations.

However, the aggregators used in PATE and PATE-GAN are not suitable for aggregating gradient vectors because they are only applicable to categorical data. Therefore, we propose a differentially private gradient aggregator (`DPGradAgg`) based on PATE. With gradient discretization, we convert gradient aggregation into a voting problem and get the noisy aggregation of teachers' votes using PATE. Additionally, we use random projection to reduce the dimension of vectors on which the aggregation is performed. The combination of these two approaches allows G-PATE to generate synthetic samples with higher data utility, even for large scale image datasets, which is hard to be achieved by PATE-GAN. The procedure is formally presented in Algorithm 7.2.

Gradient Discretization. Since PATE is originally designed for aggregating the teacher models' votes on the correct class label of an example, the aggregation mechanism in PATE only applies to categorical data. Therefore, we design a three-step algorithm to apply PATE on continuous gradient vectors. First, we discretize the gradient vector by creating a histogram and mapping each element to the midpoint of the bin it belongs to. Then, instead of voting for the class labels as in PATE, a teacher discriminator votes for k bins associated with k elements in its gradient vector. Finally, for each dimension, we calculate the bin with most votes using the `Confident-GNMax` aggregator [146]. The aggregated gradient vector consists of the midpoints of the selected bins.

With gradient discretization, the teacher discriminators can directly communicate with the student generator using the PATE mechanism. Since these teacher discriminators are trained on real data, they can provide much better guidance to the generator compared to the student discriminator in PATE-GAN, which is only trained on synthetic samples. Moreover, the `Confident-GNMax` aggregator ensures that the student generator would only improve its output in the direction agreed by most of the teacher discriminators.

Random Projection. Aggregation of high dimensional vectors is expensive in terms of privacy budget because private voting needs to be performed on each dimension of the vectors. To save privacy budget, we use random projection [147] to reduce the dimensionality of gradient vectors. Before the aggregation, we generate a random projection matrix with each component randomly drawn from a Gaussian distribution. We then project the gradient vector into a lower dimensional space using the random projection matrix. After the aggregation, the aggregated gradient vector is projected back to its original dimensions. Since the generation of random projection matrix is data-independent. It does not consume any privacy budgets.

Random projection is shown to be especially effective on image datasets. Since different pix-

Algorithm 7.2 - Differentially Private Gradient Aggregator (`DPGradAgg`).

Require: gradient vectors $\{\Delta x^{(1)}, \Delta x^{(2)}, \ldots, \Delta x^{(n)}\}$, gradient clipping constant c, number of bins B, projected dimension k ,noise parameters σ_1 and σ_2, threshold T

$k_0 \leftarrow$ the dimension of $\Delta x^{(1)}$
$R \leftarrow$ a random projection matrix of size (k_0, k) with each component randomly drawn from $\mathcal{N}(0, \frac{1}{k})$
$\{\Delta u^{(1)}, \Delta u^{(2)}, \ldots, \Delta u^{(n)}\} \leftarrow \{\Delta x^{(1)}R, \Delta x^{(2)}R, \ldots, \Delta x^{(n)}R\}$
$\Delta u \leftarrow$ empty list
for $j \in 1, 2, \ldots, k$ **do**
    $v \leftarrow$ a vector containing the jth element of all gradients in $\{\Delta u^{(1)}, \Delta u^{(2)}, \ldots, \Delta u^{(n)}\}$
    Clip v to $(-c, c)$
    $h \leftarrow$ the histogram of v with B bins of width $\frac{2c}{B}$
    $j \leftarrow$ `Confident-GNMax`$(h, \sigma_1, \sigma_2, T)$
    Append the midpoint of the j-th bin to Δu
end for
$\Delta x \leftarrow \Delta u R^T$
return Δx

els of an image are often highly correlated, the intrinsic dimension of an image is usually much lower than the number of pixels [148]. Therefore, random projection maximizes the amount of information a student generator can get from a single query to the `Confident-GNMax` aggregator, and makes it possible for G-PATE to retain reasonable utility even on high dimensional data. Moreover, random projection preserves similar squared Euclidean distance between high-dimensional vectors, therefore is beneficial to privacy protection both theoretically and empirically [149].

7.2 THEORETICAL GUARANTEES

In this section, we provide theoretical guarantees on the privacy properties for G-PATE. To start with, we propose the following definition for a differentially private data generative model.

Definition 7.1 (Differentially Private Generative Model)**.** Let G be a generative model that maps a set of points Z in the noise space $\mathcal{Z}$ to a set of records X in the data space $\mathcal{X}$. Let $\mathcal{D}$ be the training dataset of G and $\mathcal{A} : \mathcal{D} \mapsto G$ be the training algorithm. We say that G is a (ε, δ)-*differentially private data generative model* if the training algorithm $\mathcal{A}$ is (ε, δ)-differentially private.

Definition 7.1 relaxes the definition of a DP-GAN by focusing the protection only on the generative model in a GAN. This relaxation saves privacy budget during training and

improves the utility of the model. Moreover, the relaxation does not compromise the privacy guarantee for the synthetic data.

Theorem 7.1 shows that a differentially private generative model is able to support infinite number of queries to the data generator and can be used to generate multiple synthetic datasets.

Theorem 7.1. *Let G be an (ε, δ)-differentially private data generative model trained on a private dataset $\mathcal{D}$. For any $Z \in \mathcal{Z}$, the synthetic dataset $X = G(Z)$ is (ε, δ)-differentially private.*

Theorem 7.1 is a consequence of the post-processing property of differential privacy. First, the random points Z are independent of the private dataset $\mathcal{D}$. Second, one does not need to query the discriminator during the data generation process. Therefore, the synthetic dataset is generated by post-processing G and is guaranteed to be (ε, δ)-differentially private.

Next, we justify the privacy guarantee of the G-PATE method. The following lemma justifies RDP of the gradient aggregator (Algorithm 7.2).

Lemma 7.2 (Rényi Differential Privacy of `DPGradAgg`)**.** *The output of the gradient aggregator (`DPGradAgg`) proposed in Algorithm 7.2 satisfies $\left(\lambda, \sum_{1 \leq j \leq k} \varepsilon_j\right)$-RDP, where $\lambda > 1$ and ε_j is the data-dependent RDP budget with order λ for the `Confident-GNMax` aggregator on the j-th projected dimension.*

Lemma 7.2 can be proved by combining the RDP guarantee of the `Confident-GNMax` aggregator and the post-processing property of RDP. We first divide the input of `DPGradAgg` into two categories. The first category contains data independent parameters, including the gradient clipping constant c, the number of bins B, the projected dimension k, the noise parameters σ_1 and σ_2, and the threshold T. These parameters do not contain private information. The second category contains the gradient vectors $\Delta\mathbf{X} = (\Delta\mathbf{x}^{(\mathbf{1})}, \ldots, \Delta\mathbf{x}^{(\mathbf{n})})$, which are data-dependent and sensitive. Our privacy analysis focuses on the computation on $\Delta\mathbf{X}$. With random projection and gradient discretization, we convert $\Delta\mathbf{X}$ into k histograms and pass the histograms into the `Confident-GNMax` aggregator. Since `Confident-GNMax` satisfies data-dependent RDP [146], the privacy guarantee nicely propagates to the output of `DPGradAgg`.

We analyze the RDP guarantee of `DPGradAgg` by composing the privacy budget consumed by the `Confident-GNMax` aggregator on each projection dimension. Therefore, the Rényi differential privacy budget of the training algorithm is a composition of the data-dependent Rényi differential privacy budget of the `Confident-GNMax` aggregator over k dimensions. The data-dependent privacy budget for each `Confident-GNMax` aggregation is dependent on σ_1,

σ_2, and threshold T. The remaining parameters (e.g. gradient clipping constant c, number of bins B) do not influence the privacy guarantee.

The next theorem justifies RDP of the G-PATE training process.

Theorem 7.3 (Rényi Differential Privacy of G-PATE)**.** *Let $\mathcal{A}$ be the training algorithm for the student generator (Algorithm 7.1) with N training iterations and k projected dimensions. The data-dependent Rényi differential privacy for $\mathcal{A}$ with order $\lambda > 1$ is $\varepsilon = \sum_{1\leq i\leq N}\left(\sum_{1\leq j\leq k}\varepsilon_{i,j}\right)$, where $\varepsilon_{i,j}$ is the data-dependent Rényi differential privacy for the `Confident-GNMax` aggregator in the i-th iteration on the j-th projected dimension.*

For the convenience of privacy analysis, we divide the each iteration in Algorithm 7.1 into three phases: pre-processing, private computation and aggregation, and post-processing. In the pre-processing phase, the generator produces fake samples without accessing the private data. In the private computation and aggregation phase, the teacher discriminators are updated based on private data. Each teacher discriminator also generates a gradient vector. These vectors are aggregated using the `DPGradAgg` algorithm. Based on Lemma 7.2, the data-dependent RDP for this phase is $\sum_{1\leq j\leq k}\varepsilon_{i,j}$. In the post-processing phase, the student generator is updated using the privately-aggregated gradient vector $\Delta\mathbf{x_j}^{\text{priv}}$. It satisfies RDP because of the post-processing property. Finally, the RDP of Algorithm 7.1 is composed over N training iterations.

The next theorem provides a theoretical guarantee on the differential privacy of the G-PATE method.

Theorem 7.4 (Differential Privacy of G-PATE)**.** *Given a sensitive dataset $\mathcal{D}$ and parameters $0 < \delta < 1$, let G be the student generator trained by Algorithm 7.1. There exists $\varepsilon > 0$ and $\lambda > 1$ so that G is a $(\varepsilon + \frac{\log 1/\delta}{\lambda-1}, \delta)$-differentially private data generative model.*

Theorem 7.4 is the consequence of converting the Rényi differential privacy guarantee in Theorem 7.3 to differential privacy (Theorem 2.1).

7.3 EXPERIMENTAL EVALUATION

We evaluate G-PATE against two state-of-the-art benchmarks: DP-GAN and PATE-GAN. To compare the performance of different data generators, we train a classifier on the synthetic data and test it on the original data to benchmark the quality of the synthetic data.

We first perform comparative analysis with PATE-GAN and DP-GAN on the datasets used in the corresponding works (i.e., Kaggle credit dataset and MNIST dataset). In addition, we evaluate G-PATE on Fashion-MNIST and more privacy sensitive large scale face dataset CelebA.

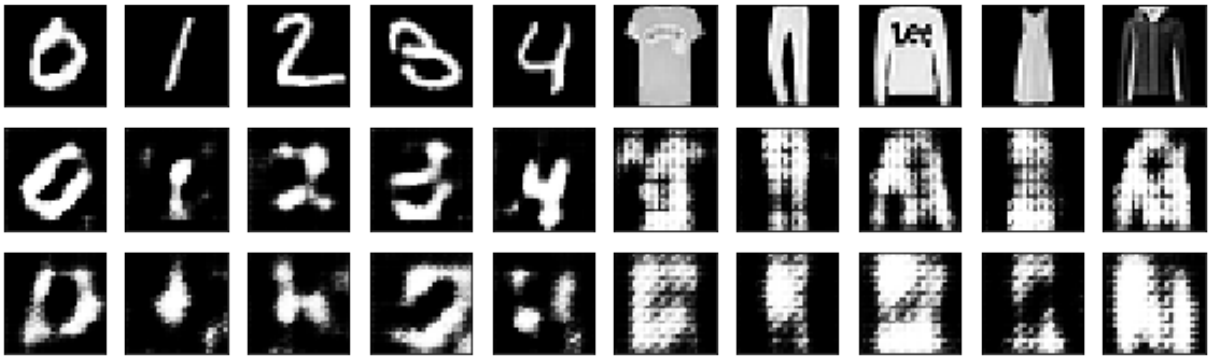


Figure 7.2: **Visualization of generated instances by G-PATE.** Row 1 (real image), row 2 ($\varepsilon = 10, \delta = 10^{-5}$) and row 3 ($\varepsilon = 1, \delta = 10^{-5}$) each presents one image from each class (the left 5 columns are MNIST images, and the right 5 columns are Fashion-MNIST images). When $\varepsilon = 1$, G-PATE does not generate high-quality images. However, it preserves partial features in the training images, so the synthetic images are useful to preserve data utility which can be seen from our quantitative results.

7.3.1 Experimental Setup

To compare with PATE-GAN, we use the same Kaggle credit card fraud detection dataset [150] (Kaggle Credit) as in [37]. The dataset contains 284,807 samples representing transactions made by European cardholders' credit cards in September 2013, and 492 (0.2%) of these samples are fraudulent transactions. Each sample consists of 29 continuous features which are the results of a PCA transformation from the original features.

To demonstrate the superiority of G-PATE to PATE-GAN on high dimensional image datasets, we train G-PATE on MNIST, Fashion-MNIST [151], and the celebrity face datasets CelebA [108]. MNIST and Fashion-MNIST consist of 60,000 training examples and 10,000 testing examples. Each example is a 28×28 grayscale image, associated with a label from 10 classes. The CelebA dataset contains 202,599 images aligned and cropped based on the human face. We create three datasets: CelebA-Gender(S) is a binary classification dataset that uses the gender attributes as the labels and resizes the images to 32x32x3; to demonstrate the scalability of G-PATE we also create CelebA-Gender(L) with the same label while resizing the images to 64x64x3; CelebA-Hair contains images as 64x64x3 with three hair color attributes (black/blonde/brown). We follow the official training and testing partition as [108].

For the Kaggle Credit dataset, both the generator and discriminator networks of G-PATE are fully connected neural network with the same architecture as PATE-GAN [37]. We use random projection with 5 projected dimensions during gradient aggregation. We use the DCGAN [152] structure on the image datasets. We use random projection with 10 projected dimensions during gradient aggregation.

	DC-GAN	PATE-GAN	DP-GAN	G-PATE
LR	0.9430	0.8728	0.8720	**0.9251**
AdaBoost	0.9416	0.8959	0.8809	**0.8981**
Bagging	0.9379	0.8877	0.8657	**0.8964**
MLP	0.9444	0.8925	0.8787	**0.9093**
Average	0.9417	0.8872	0.8743	**0.9072**

Table 7.1: **Performance Comparison on Kaggle Credit Dataset.** The table presents AUROC of the classifier trained on synthetic data and tested on real data. The performance is evaluated over 4 different classifiers: logistic regression (LR), AdaBoost, bagging, and multi-layer perceptron (MLP). PATE-GAN, DP-GAN, and G-PATE all satisfy $(1, 10^{-5})$-differential privacy. Vanilla DC-GAN has no privacy protection. The best results out of different models are highlighted in bold.

	Projection Dimensions				# of Teachers		
	5	10	20	No Projection	2000	3000	**4000**
MNIST	0.4638	**0.5810**	0.5604	0.1141	0.4240	0.5218	**0.5810**
Fashion	0.5129	**0.5567**	0.5172	0.1268	0.3997	0.4874	**0.5567**

Table 7.2: **Analysis on the Number of Teachers and the Projection Dimensions.** We performed comprehensive studies on the number of teachers and the the projection dimensions on MNIST with $\varepsilon = 1$ and $\delta = 10^{-5}$. The model has the best performance with 4000 teacher models and projection dimension equals to 10.

7.3.2 Comparison with DP-GAN and PATE-GAN

Kaggle Credit. The Kaggle Credit dataset is highly unbalanced. In PATE-GAN, the ratio between positive and negative classes in the sensitive training set is assumed to be public information. In contrast, the G-PATE method does not rely on any public information about the sensitive training dataset. It calculates the ratio between positive and negative classes using Laplacian mechanism [38] with $\varepsilon = 0.01$. We then train a $(0.99, 10^{-5})$-differentially private data generator and sample the synthetic records according to the noisy class ratios. By the composition theorem of differential privacy [38], the data generation mechanism is $(1, 10^{-5})$-differentially private.

To compare with PATE-GAN, we select 4 commonly used classifiers evaluated in [37]. The performance of a generator is measured by the AUROC of the 4 classifiers trained on the corresponding synthetic data. We evaluate G-PATE under the same experimental setups as PATE-GAN for $\varepsilon = 1$. The results for PATE-GAN and DP-GAN are from [37], and we get a

Dataset	DC-GAN		DP-GAN	G-PATE
MNIST	0.9653 ($\varepsilon = \infty$)	$\varepsilon = 1$	0.4036	**0.5810**
		$\varepsilon = 10$	0.8011	**0.8092**
Fashion-MNIST	0.8032 ($\varepsilon = \infty$)	$\varepsilon = 1$	0.1053	**0.5567**
		$\varepsilon = 10$	0.6098	**0.6934**
CelebA-Gender(S)	0.8002 ($\varepsilon = \infty$)	$\varepsilon = 1$	0.5201	**0.7016**
		$\varepsilon = 10$	0.5409	**0.7072**
CelebA-Gender(L)	0.8149 ($\varepsilon = \infty$)	$\varepsilon = 1$	0.5330	**0.6702**
		$\varepsilon = 10$	0.5211	**0.6897**
CelebA-Hair	0.7678 ($\varepsilon = \infty$)	$\varepsilon = 1$	0.3447	**0.4985**
		$\varepsilon = 10$	0.3920	**0.6217**

Table 7.3: **Performance on Image Datasets.** We compare G-PATE with DP-GAN and vanilla DC-GAN. The table presents the classification accuracy of a model trained on the generated data and tested on real data. DP-GAN and G-PATE are both evaluated under two private settings: $\varepsilon = 1, \delta = 10^{-5}$ and $\varepsilon = 10, \delta = 10^{-5}$.

higher baseline performance from DC-GAN compared to the baseline performance reported in [37].

Table 7.1 presents the comparative analysis between G-PATE and PATE-GAN on Kaggle Credit dataset. G-PATE outperforms both PATE-GAN and DP-GAN and is close to the vanilla DC-GAN which has no privacy protection. The good performance of G-PATE is partly due to the relatively low dimensionality of the Kaggle Credit dataset and the abundance of training examples.

MNIST and Fashion-MNIST. To understand G-PATE's performance on image datasets, we perform comparative analysis between G-PATE and DP-GAN on the MNIST and Fashion-MNIST datasets. We evaluate the generator by the 10-class classification accuracy of models trained on synthetic data and tested on real data (Table 7.3). The analysis is performed under two performance settings: $\varepsilon = 1, \delta = 10^{-5}$ and $\varepsilon = 10, \delta = 10^{-5}$. G-PATE outperforms DP-GAN under both settings, and there is a more significant improvement for the setting with stronger privacy guarantee (i.e., $\varepsilon = 1$). Specifically, we observe that DP-GAN fails to converge on the Fashion-MNIST dataset with $\varepsilon = 1$. The synthetic records generated by DP-GAN under this setting are close to random noise while the model trained on G-PATE generated data retains an accuracy of 55.67%.

CelebA. To demonstrate the scalability of our algorithm, we evaluate G-PATE and DP-GAN to compare the data utility. As shown in Table 7.3, the synthetic data generated by

ε	MNIST		Fashion-MNIST	
	DP-GAN	G-PATE	DP-GAN	G-PATE
0.2	0.1104	**0.2230**	0.1021	**0.1874**
0.4	0.1524	**0.2478**	0.1302	**0.3020**
0.6	0.1022	**0.4184**	0.0998	**0.4283**
0.8	0.3732	**0.5377**	0.1210	**0.5258**
1.0	0.4046	**0.5801**	0.1053	**0.5567**

Table 7.4: **Performance Comparison on Image Datasets given small privacy budgets.** DP-GAN and G-PATE are both evaluated in the same way as Table 7.3 given $\delta = 10^{-5}$ and low $\varepsilon \leq 1.0$.

G-PATE is highly utility-preserving, while DP-GAN can barely converge given the high-dimensionality of the data. In particular, even with the rigorous privacy budget $\varepsilon = 1$, the accuracy of the generated data by G-PATE on CelebA-Gender(S) is only around 10% lower than the vanilla DC-GAN. It is also worth noticing that although the dimensionality of CelebA-Gender(L) is 4x larger than CelebA-Gender(S), the utility of both datasets are very close, which again demonstrates the scalability of G-PATE.

Analysis on the Number of Teachers and the Projection Dimensions. We perform comprehensive ablation studies on the number of teachers and the the projection dimensions to gain better understanding about G-PATE. As shown in Table 7.2, G-PATE benefits from having more teacher discriminators. Under the same privacy guarantee, the number of noisy votes (σ_1 and σ_2) remains the same, so the output of the noisy voting algorithm is more likely to be correct, and the model would get better performance. However, this benefit diminishes as the training set for each teacher model gets smaller with the increasing number of teachers, and 4000 teachers have already achieved satisfiable results. Table 7.2 also demonstrates the effectiveness of the random projection method, which improves the classification accuracy by around 0.45.

Analysis under Limited Privacy Budgets. We conduct another set of ablation studies given limited privacy budgets. From Table 7.4, we can observe that our algorithm starts to converge even under small ε on both MNIST and Fashion-MNIST datasets. The utility stably increases when the privacy budgets increase. Among different small ε, G-PATE achieves significantly higher accuracy than DP-GAN. In particular, the accuracy of G-PATE under $\varepsilon = 0.6$ is four times higher than DP-GAN, which indicates that G-PATE is able to generate differentially private data with high utility even under low privacy budgets.

7.4 CONCLUSION

This chapter proposes G-PATE, a novel approach for training a differentially private data generator by ensuring differential privacy property on the information flow from the discriminator to generator in GAN. G-PATE is enabled by a novel differentially private gradient aggregation mechanism combined with random projection. It significantly outperforms prior work on both image and non-image datasets in terms of preserving data utility for generated datasets. Moreover, G-PATE is able to preserve reasonable data utility on complex high-dimensional image dataset for which DP-GAN can hardly converge. Beyond the high utility compared with the state-of-the-art differentially private data generative models (DP-GAN), G-PATE is shown to be able to preserve high data utility even under small privacy budgets.

CHAPTER 8: DIFFERENTIALLY PRIVATE GRAPH CONVOLUTIONAL NEURAL NETWORKS

In this chapter, we propose the first differentially private GCN (DP-GCN) based on node clustering. We show that directly adding Laplacian noise in the adjacency matrix is insufficient in preserving model utility because the sparse signals in an adjacency matrix are highly vulnerable to noise injection. Motivated by this, we design a noise-resilient GCN structure via node clustering. Our DP-GCN relies on aggregated node-to-cluster connections that are less sensitive to the fine-grained noise. We prove that DP-GCN guarantees edge differential privacy and empirically show that it preserves much higher data accuracy compared to different baselines on several large graph datasets. In addition, we gain insights about factors that affect edge privacy on graph data by analyzing the tradeoff between data utility and node degree distribution as well as the number of node clusters. This chapter is based on joint work with Fan Wu, Bhavya Kailkhura, Qian Chen, and Bo Li.

Graphical neural networks (GCNs) have been widely applied to different applications based on structured graph datasets [153, 154]. However, machine learning models including GCNs today are mostly a privilege of large datasets with rich statistics, which would potentially contain privacy sensitive information. For instance, a social network graph is important for analyzing and discovering various social issues, including disease transmission, emotional contagion, and occupational mobility. However, the edges in social networks may reflect the communication, the price of commercial trade, or the intimacy of relationship. As a result, protecting the privacy on such graph data is very of great importance.

Existing work has provided different privacy mechanisms to protect edge privacy in traditional graph analysis [155, 156, 157]. However, so far there is no work on successfully protecting the edge privacy in the context of GCNs despite their wide applications. Driven by this, in this chapter we mainly want to understand two questions: *(1) Is standard Laplacian noise mechanism enough to protect the edge privacy as well as retain reasonable data utility for GCNs? (2) What could be done to improve the data utility without breaking the edge privacy promise?*

In particular, we conduct a series of experiments and draw the observation that only adding standard Laplacian noise to the adjacency matrix of graphs for GCNs will largely affect the data utility and almost renders it to be useless. To address this concern, we propose to leverage the node information such as node labels and node features to perform node clustering, aiming to identify less important edges and eliminate them within the finite privacy budget. We then perform edge reconstruction on the node cluster level, indicating

the graph connectivity between nodes and node clusters. Together with our theoretic analysis for node clustering and differential privacy, we prove that our proposed DP-GCN is able to guarantee differential privacy for edges, and we further provide intuition on its implications. Besides, we look into different node clustering strategies to evaluate their impacts on data utility for DP-GCN, including random, hierarchy, and only node-label or node-feature based approaches.

Empirically, we conduct extensive experiments on graphs including Cora, Citeseer [158], and the Facebook Large Page-Page Network [159]. We show that the proposed DP-GCN is able to largely protect the data utility and outperforms other baselines significantly. A range of ablation studies are also evaluated and draw interesting conclusions which would shed light on future research for protecting graph privacy for GCNs. For example, we observe that DP-GCN has relatively better performance on high-degree nodes because they are more robust to noise injection on edges.

Technical Contributions In this chapter, we take the first step towards protecting edge privacy by training differentially private GCNs and leveraging different graph clustering strategies. We make contributions in both the theoretical and empirical aspects.

- To our best knowledge, we propose the first differentially private graphical neural networks (DP-GCN) based on a simple yet effective hierarchical node clustering strategy with corresponding theoretic analysis.
- We show that directly adding Laplacian noise to the adjacency matrix is insufficient in training a differentially private GCN with high utility, which provides interesting insights into the preservation of edge privacy in GCNs.
- We prove that the trained DP-GCN model is edge differentially private, and we analyze the effects that different node clustering approaches have on data utility.
- We conduct extensive experiments on large scale graph datasets including Cora, Citeseer [158], and the Facebook Large Page-Page Network [159]. We show that the proposed DP-GCN outperforms other baselines in terms of the graph data utility. Even with small privacy budget $\varepsilon = 0.1$, the proposed DP-GCN is able to achieve performance close to the non-private vanilla GCN. For instance, on the Facebook dataset, the accuracy of DP-GCN only about 6.35% lower than that of vanilla GCN.

8.1 DIFFERENTIALLY PRIVATE GCN

We first discuss the limitations of directly applying the Laplace mechanism to motivate the proposed approach. Specifically, we show that, due to the sparsity of the adjacency

matrix, directly adding Laplacian noise would greatly distort the structural information in the original graph.

Suppose A is the adjacency matrix of a graph. Because A is symmetric, adding/dropping an edge in the graph would result in two entries in A to change by 1. Therefore, the ℓ_1 sensitivity of A equals to 2. To ensure ε-edge differential privacy, we need to add independent Laplacian noise with $\lambda = 2/\varepsilon$ on each entry in A.

For example, when $\varepsilon = 1$, we sample a random variable $Z \sim \text{Laplace}(0, 2)$, compute $A + Z$, and clip the output to $[0, 1]$ to get a legitimate noisy adjacency matrix. Based on the c.d.f. of the Laplace distribution, $\Pr[Z \leq -1] = \Pr[Z \geq 1] = 0.3$. Consequently, after noise injection, each non-zero entry in A has at least 30% chance of becoming 0, and each zero entry in A has at least 30% chance of becoming 1. The noise injection process is equivalent to randomly dropping edges with probability of 0.3 and adding edges with probability of 0.3 between *all* the disconnected node pairs. This process is a huge distortion of the original graph structure.

8.1.1 Graph Aggregation via Node Clustering

The poor performance of directly injecting Laplacian noise can mainly be attributed to the sparsity of A and the small value of each entry in A. To solve this problem, we need to aggregate the graph structural information to construct a denser matrix with higher values. Moreover, the aggregation method should maintain a low sensitivity so that adding/dropping a single edge has little effect on the aggregation results. In this section, we design a cluster adjacency matrix to aggregate high-level structural information of a graph.

Definition 8.1 (Cluster Adjacency Matrix A_c)**.** Given an undirected graph with N nodes and M non-overlapping node clusters, the cluster adjacency matrix A_c is an $N \times M$ matrix where each element $\hat{a}_{ij}$ equals to the number of edges between node i and cluster j.

The cluster adjacency matrix aggregates node-to-node connection information to the node-to-cluster level. Empirically, this aggregation provides better edge privacy protection because each single edge plays a less important role in the aggregation results. Theoretically, we prove that the cluster adjacency matrix has the same L_1 sensitivity as the node adjacency matrix, therefore the amount of injected noise to achieve differential privacy is the same between the two matrices. However, since the cluster adjacency matrix has higher values and lower dimensions, it is more robust to the injection of the same level of noise.

The following theorem calculates the ℓ_1 sensitivity of a cluster adjacency matrix.

Theorem 8.1. *The ℓ_1 sensitivity of a cluster adjacency matrix A_c is 2.*

8.1.2 Noise-Resilient GCN Structure

To take advantage of the noise-resilient property of the cluster adjacency matrix, we redesign the structure of a deep convolutional layer by replacing the node adjacency matrix A with the cluster adjacency matrix A_c. Specifically, given a cluster adjacency matrix of size $N \times M$, we design the following projection matrix P of size $M \times N$:

$$p_{ji} = \begin{cases} 1/|C_j|, & \text{if } v_i \in C_j, \\ 0, & \text{otherwise,} \end{cases} \tag{8.1}$$

where C_j represents cluster j and v_i represents node i ($1 \leq i \leq N, 1 \leq j \leq M$).

The projection matrix P maps the node-to-cluster connection back to node-to-node connection by equally dividing the number of edges among all the nodes within the same cluster. For example, suppose node v has 10 edges connected to a cluster C with 100 nodes (i.e., $|C| = 100$). After the projection, v is connected to all the nodes in C with edge weight equals to 0.1.

By combining the cluster adjacency matrix A_c and the projection matrix P, we modify each graph convolutional layer as follows:

$$H^{(l+1)} = \sigma\left((D^{-1}A_cP + I_N)H^{(l)}W\right), \tag{8.2}$$

where A_c is the cluster adjacency matrix, D is the degree matrix based on Ac, P is the projection matrix, $H^{(l)}$ is the output of the previous layer, W is the trainable weight matrix, and σ is the activation function. We use the identity matrix I_N to add self-connections on each node.

The noise-resilient GCN structure introduces two adaptations to improve privacy protection and enhance model robustness against noise injection. First, instead of using the fine-grained node-to-node connection information, the model takes advantage of the graph connectivity between a node and a node cluster. This aggregation prevents the model from learning information about individual edges, therefore provides better privacy protection. Second, we add an additional identity matrix I_N to ensure that the self-connection is not lost during the aggregation. Unlike a traditional GCN, in our proposed structure, I_N is not normalized by node degree. This modification improves the importance of self-connections enhances the model's robustness against noise injection on edges.

8.1.3 Differentially Private GCN

Algorithm 8.1 represents the training process of a DP-GCN. The algorithm uses the GCN structure in E.q.(8.2) and ensures edge differential privacy by adding Laplacian noise to the cluster adjacency matrix A_c.

Algorithm 8.1 Training process of DPGCN.

Require: Cluster adjacency matrix A_c, projection matrix P, node feature matrix X, training labels Y_{train}, privacy budget ε
Ensure: The DP-GCN model
1: Sample laplacian noise $Z \sim \text{Laplace}(0, 2/\varepsilon)$ with the same size as A_c
2: Add noise to the cluster adjacency matrix $A'_c \leftarrow A_c + Z$
3: Train a DP-GCN using $A'_c, P, X, Y_{\text{train}}$

The following theorem shows DP guarantee of Algorithm 8.1.

Theorem 8.2. *The DP-GCN model trained by Algorithm 8.1 is ε-edge differentially private.*

Algorithm 8.2 Hierarchical Node Clustering.

Require: Features and labels of training nodes X_{train}, Y_{train}; features of unlabeled nodes X_{test}; number of classes K;
Ensure: Node clusters $\{C_1, C_2, \ldots, C_M\}$
1: **for** X_i in $X_{\text{unlabeled}}$ **do**
2: Find the k instances in the training set that are closest to X_i in the feature space
3: $y_i \leftarrow$ the majority voting result of the labels of the k instances
4: **end for**{Label Bootstrapping}
5: **for** i from 1 to K **do**
6: $\hat{C}_i \leftarrow$ the set of instances that have label i
7: **end for**{Label-Based Clustering}
8: Cluster size $s \leftarrow \min_i |\hat{C}_i|$
9: **for** i from 1 to K **do**
10: Number of sub-clusters $m_i \leftarrow |\hat{C}_i|/s$
11: $centers \leftarrow$ K-Means$(\hat{C}_i, m_i)$
12: Assign a nearly equal number of instances to each $centers$ based on the distance in the feature space
13: **for** each node c in $centers$ **do**
14: Group nodes assigned to c as a new cluster
15: **end for**
16: **end for**{Feature-Based Sub-Clustering}

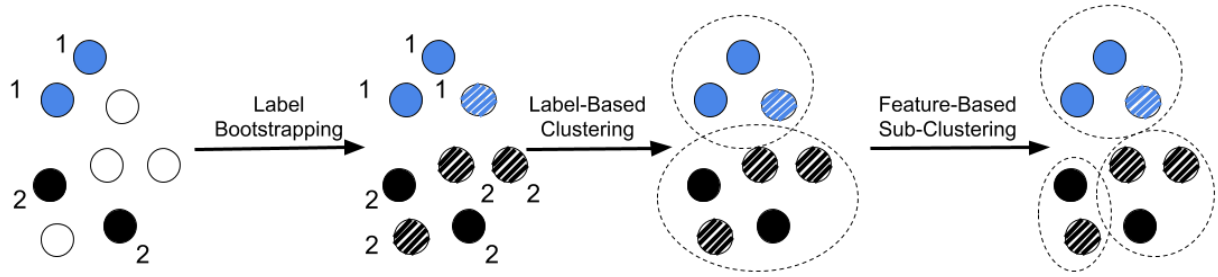


Figure 8.1: **Overview of the Hierarchical Clustering Algorithm**. (1) Label Bootstrapping: assign labels to test nodes based on their neighbors in the feature space; (2) Label-Based Clustering: group nodes into high-level clusters using labels; (3) Feature-Based Sub-Clustering: divide the high-level clusters into sub-clusters using node features.

8.1.4 Hierarchical Node Clustering

While the cluster adjacency matrix improves privacy protection by aggregating the edges, it brings a trade-off in the performance of DP-GCN due to the loss of fine-grained edge information. Therefore, choosing an appropriate clustering algorithm is an important factor in striking a balance between privacy protection and model performance.

In DP-GCN, the number of edges connected to a cluster is evenly divided among the nodes in that cluster. Consequently, for each deep convolutional layer, the output $H^{(l+1)}$ would be a good approximation of the output of an original deep convolutional layer, only if nodes in the same cluster share similar embedding in the prior layer output $H^{(l)}$. However, clustering via node embedding is not plausible because the embedding is only accessible after training the model.

In this section, we introduce a hierarchical node clustering algorithm to divide nodes into non-overlappig clusters so that nodes in the same cluster are likely to have similar node embedding in a GCN. This clustering problem is essentially different from traditional graph clustering problems because the embedding produced by a graph convolutional layer is a combination of graph structure, node features, and training labels. Hence, node clustering algorithms that solely rely on the graph structures, such as Metis [160], are not suitable.

Besides combining different signals in the clustering algorithm, we also need to take consideration of the privacy cost of accessing the graph structure. On the one hand, clustering based on the graph structures without privacy protection may leak the exact sensitive information we aim to protect with edge differential privacy. On the other hand, extracting useful graph statistics with edge differential privacy can consume the precious privacy budget that is crucial to training an accurate DP-GCN model. Therefore, we design a *graph-agnostic*

node clustering algorithm to approximate the GCN node embedding based on node features and labels. We show that the algorithm improves the performance of DP-GCN with *no extra privacy cost*.

The key of the clustering algorithm is a hierarchical approach combining node features and labels. Although both features and labels provide valuable information to a GCN, they play different roles in the training process. Ideally, the embedding produced by a graph convolutional layer should separate nodes with different labels to make node classification easier. To preserve this trend in our clustering algorithm, in the highest clustering hierarchy, we use label-based clustering method to ensure that training nodes with different labels do not belong to the same cluster. In addition, we utilize node features to further divide the huge label clusters into smaller sub-clusters with approximately the same size. Specifically, the clustering algorithm contains three steps: label bootstrapping, label-based clustering, and feature-based sub-clustering. Algorithm 8.2 presents the pseudo-code for the clustering process.

Label Bootstrapping. In the semi-supervised node classification setting, the training graph consists of both labeled training nodes and unlabeled testing nodes. To start with, we need to assign a bootstrap label to the testing nodes in preparation of label-based clustering. Since, in most cases, training nodes only consist of a small portion of the graph, We use k nearest neighbor classification algorithm to assign a bootstrap label for each testing nodes. To avoid accessing the sensitive graph structure, the neighbors are selected using the Euclidean distance of the feature vectors.

Label-Based Clustering. Once we obtain the bootstrapping labels, we run a label-based clustering algorithm to group nodes with the same label into the same cluster. This process results in K node clusters, where K is the number of node classes.

Feature-Based Sub-Clustering. However, when the node labels are unbalanced, the label-based clustering method results in clusters with unequal sizes, which can make predictions biased towards the majority class. To solve this problem, we further divide the large label clusters into smaller ones using the k-means algorithm method based on node features. Each label cluster is divided into different number of sub-clusters to ensure that all sub-clusters have approximately the same size.

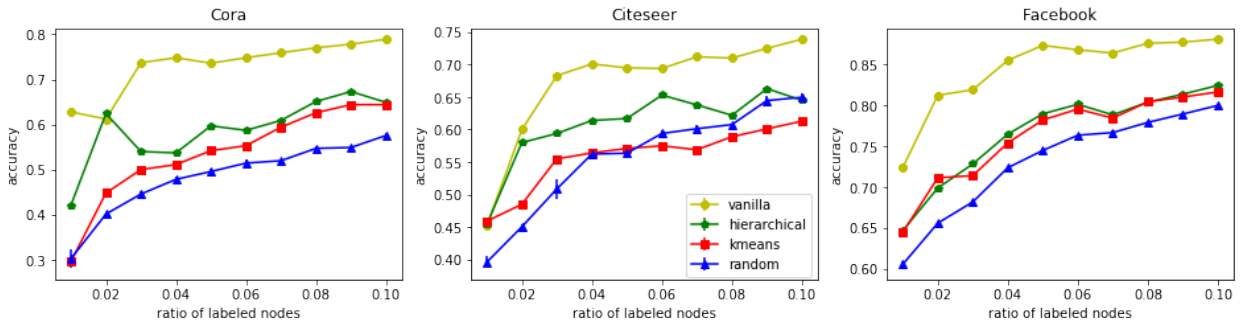


Figure 8.2: **Comparison of Node Clustering Algorithms on Clean Graphs.** We compare the performance of our hierarchical clustering algorithm against three baselines: no clustering (vanilla), k-means clustering, and random clustering. The hierarchical approach consistently outperforms k-means clustering and random clustering.

Dataset	**#Nodes**	**#Edges**	**#Classes**	**Density**
Cora	2,708	5,429	7	0.0015
Citeseer	3,327	4,732	6	0.0009
Facebook	22,470	171,002	4	0.0007

Table 8.1: Statistics of Graph Datasets

8.2 EXPERIMENTS

In this section, we evaluate the performance of the proposed hierarchical node clustering algorithm and DP-GCN on three real-life datasets under different privacy settings.

8.2.1 Datasets

The statistics of the datasets are summarized in Table 8.1. We use two citation networks, Cora and Citeseer [158] to evaluate the performance of our approach. In addition, we experiment on a social network dataset Facebook Large Page-Page Network [159], where the nodes are official Facebook pages of four categories (politicians, governmental organizations, television shows and companies) defined by Facebook, and the edges are the mutual likes between the official Facebook pages. With the natural realization that the relationship between these identities concerns privacy, here we use DP-GCN to provide our own solution to this problem.

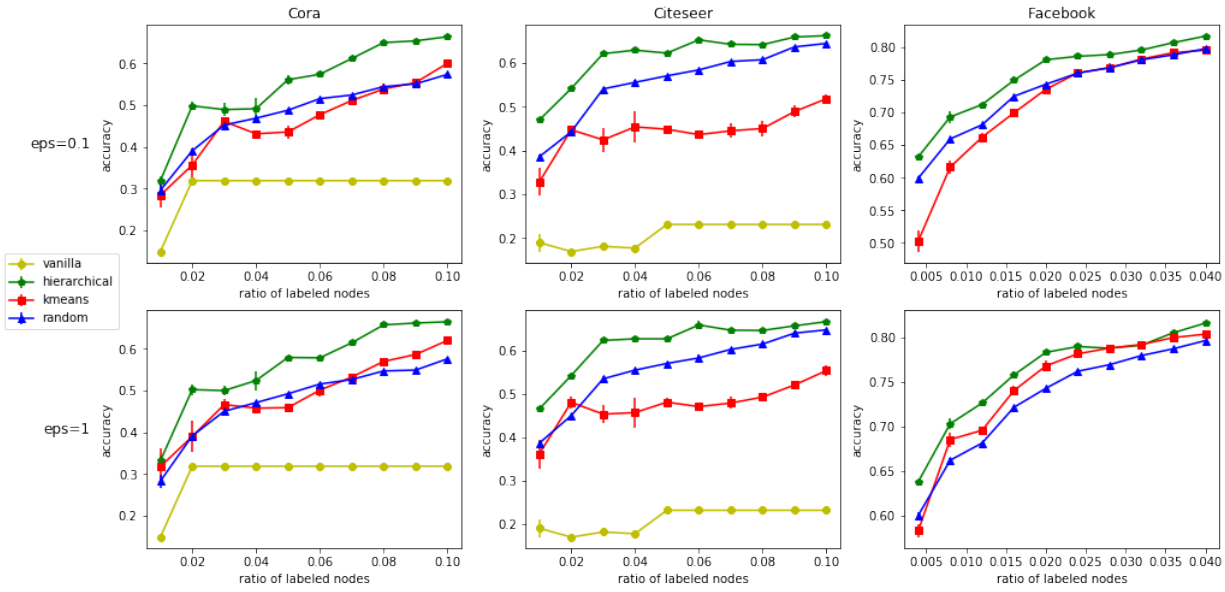


Figure 8.3: **Performance of DP-GCN under Different Clustering Algorithms.** We compare the performance of DP-GCN under different clustering algorithms. We omitted the accuracy of the vanilla approach on the Facebook dataset (0.2924) for better presentation. The cluster-based DP-GCN structure consistently outperforms the vanilla approach, and the hierarchical clustering algorithm preserves the highest utility under different privacy budget.

8.2.2 Experimental Setup

To demonstrate the generalizability of our approach, we do evaluation on a multitude of scenarios that only differ in the ratio of training instances, i.e., labeled nodes in the graph. For Cora and Citeseer, in the selection of test instances, we follow the experiment setups in [161] and use the hold-out test set of 1000 instances; we use a fixed set of size 20% of the remaining instances as the validation set, and take another subset of size from 1% to 10% as the labeled training set and use a step size of 1%. For the Facebook dataset, we set aside 10% of all instances for testing and 10% for validation. Since it is a large-scale dataset containing tremendous nodes, we limit the ratio of the labeled nodes to 0.4%~4% with step size equal to 0.4%.

For both vanilla GCN [162] and DP-GCN, we use a two layer network with the hidden layer size 16 and dropout rate 0.5. In all the experiments, we train the GCN using Adam [163] with learning rate 0.01 and weight decay 5e-4 for 500 epochs, and all experiments achieve convergence within the range. We use the same set of hyperparameters as [162] except for the number of epochs – directly adding Laplacian noise to the adjacency matrix in a vanilla GAN requires more epochs to converge. The results on the noisy graph are averaged over runs with three random seeds for noise generation.

8.2.3 Evaluation of Node Clustering.

In the first set of experiments, we aim to understand the influence of node clustering on the performance of GCNs. Specifically, we measure the performance of GCNs trained under different clustering algorithm without adding Laplacian noise to guarantee differential privacy. We compare our hierarchical node clustering with three other approaches. The vanilla GCN [162] is chosen as an ideal upper-bound, given that the information utilized by it is complete and exact. The other two baselines are designed to also suit in the same node clustering framework, but are rather "flat" without exploiting a hierarchical structure.

One baseline adopts purely random clustering, where each node is randomly assigned an initial label which denotes the cluster it belongs to. The other baseline applies the classical centroid-based clustering algorithm—k-means [164] to the new feature space to group the nodes into clusters. The new feature is designed to have a fair comparison with our hierarchical approach. In detail, for each node in the graph, its new feature is a concatenation of the original node feature with the one-hot representation of the initial label. The initial label of a labeled instance is the groundtruth label; for those unlabeled instances, the initial label is the bootstrap label derived in the same way as the label-bootstrapping process introduced previously.

Setting aside the vanilla approach, we can see in Figure 8.2 that hierarchical node clustering is the best among all the clustering-based approaches. The advantage of the vanilla approach is pronounced, which is plainly understandable given the amount of information it utilizes. The random clustering goes to another extreme as to retaining little graph structural information. Notice that the k-means clustering on the new feature space in essence utilizes the same amount of information as the hierarchical approach which sequentially exploits features and labels, whereas it fails to achieve the same good result as the hierarchical approach. This convinces us of the effectiveness of the hierarchical clustering algorithm we design.

8.2.4 Evaluation of DP-GCN

Having studied the performance of different approaches on the clean graph, we now turn to the evaluation of DP-GCN. Since DP-GCN is trained on a noisy graph, there is a risk that the model might be overfitted to the injected noise and have good performance only on the input noisy graph. Therefore, to have a fair understanding on the performance on DP-GCNs, our experiment strictly follows the procedure below. First, we inject different levels of noise ($\varepsilon \in \{0.1, 1\}$) to the clean graph according to the privacy requirement. Next, we train a differentially private GCN on the noisy graph using the same set of approaches explained in

the previous section. Finally, we test the model on the clean graph to understand the utility of the model.

The evaluation results are reported in Figure 8.3. A straightforward observation is that, with different clustering algorithms (i.e., hierarchical, k-means, and random), the new clustering-based DP-GCN structure consistently outperforms the approach of naively adding Laplacian noise on the input adjacency matrix. The score of the vanilla approach is particularly low (0.2924) on the Facebook dataset and remains the same for all ε and ratio of labeled nodes we experimented with. Therefore, we omit it in Figure 8.3 for better presentation. The detailed figures can be referred to in the supplementary materials.

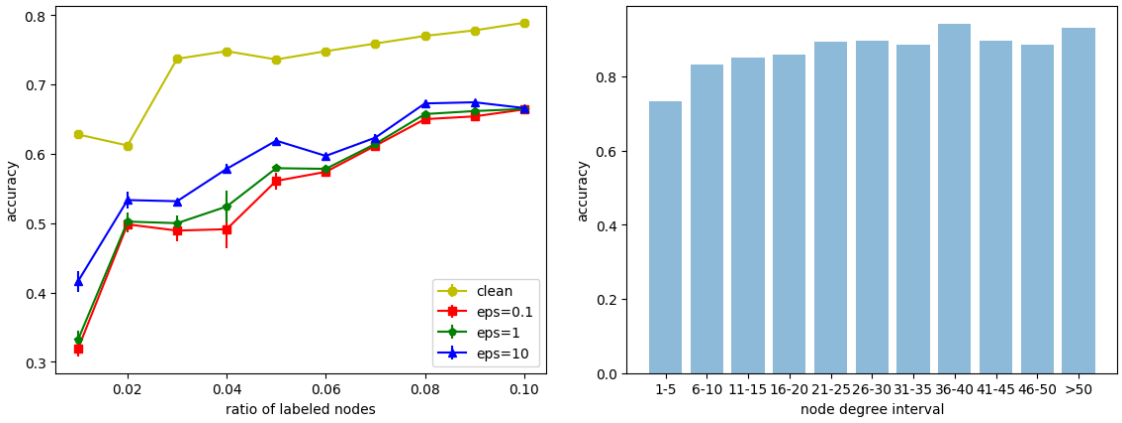
Among the three clustering-based approaches, it is noticeable that hierarchical node clustering is the invariable winner, which is also consistent with the results on clean graph in Figure 8.2. Specifically, when $\varepsilon = 0.1$, DP-GCN with hierarchical node clustering achieves an accuracy of 81.72% on Facebook dataset when 4% of the nodes are labeled, which is only 6.35% lower than the performance of vanilla non-private GCN.

8.2.5 Ablation Studies on DP-GCN

In this section, we perform ablation studies to further understand how different factors influence the performance of DP-GCN.

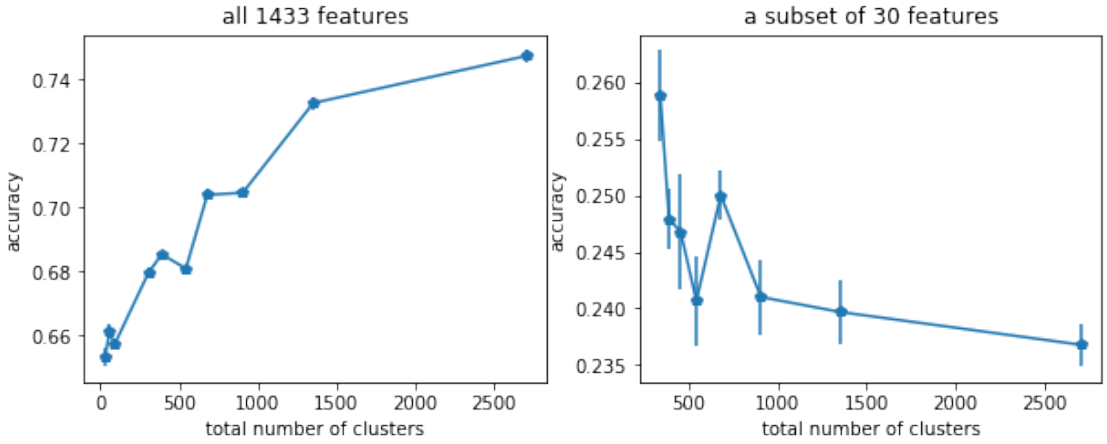
Privacy Budgets. The privacy budget ε controls the scale of the injected Laplacian noise. Generally, smaller ε provides a better privacy protection at the expense of incurring higher utility loss. We explore the utility of DP-GCN with $\varepsilon = 0.1$, 1, and 10, and compare the result with that of vanilla GCN on the clean graph where no noise is injected. As shown in Figure 8.4a, the utility drops with the decrease of privacy budget. The result is reported on Cora dataset because the performance gap between vanilla GCN on the clean graph and our DP-GCN on the noisy graph on this dataset is the largest among all three datasets we use. Even so, the model still retains an accuracy of 66.40% when $\varepsilon = 0.1$ with the gap equal to 12.5% at the point of 10% labeled nodes. On Citeseer dataset and Facebook dataset, the gap is 7.63% and 6.35% respectively when 10% and 4% of the nodes are labeled.

Node-Degree Distribution. We observe that the performance of DP-GCN varies on nodes with different degrees—the model has higher accuracy on high-degree nodes because these nodes are more robust to adding/dropping a few neighboring edges. In particular, we perform the ablation study on the Facebook dataset, which has the highest variation in node degree. Figure 8.4b shows DP-GCN's average accuracy on nodes with different degrees when



(a) Privacy Budgets.

(b) Node-Degree Distribution.



(c) Number of Node Clusters.

Figure 8.4: **Ablation Studies on DP-GCN.** We study the performance of DP-GCN under different privacy budget, node-degree distribution, and number of clusters.

$\varepsilon = 0.1$ and the ratio of labeled nodes is 4%. The difference is particularly significant for nodes with very low degree (e.g. ≤ 5). On average, DP-GCN's accuracy on these nodes is 13.27% lower than that on nodes with degree greater than 5. Since the prediction on a low-degree node heavily relies on its small number of neighbors in the graph, it is more susceptible to the adding/dropping of neighboring edges.

Number of Node Clusters. The number of node clusters controls the balance between improving privacy with aggregation and retaining useful graph connection information. In DP-GCN, adapting a higher number of node clusters can preserve more fine-grained connection information during the training process. Meanwhile, with more clusters, the influence of Laplacian noise injection increases because each entry in the cluster adjacency matrix has a smaller value. Interestingly, we observe that influence of increasing node clusters is closely

correlated to the quality of node features.

We perform the ablation study on the Cora dataset with two controlled variables: the number of node clusters and the number of node features. The results are presented in Figure 8.4c. In the first study, all 1433 node features are used for clustering, training, and training and testing, and increasing the number of clusters improves the performance of DP-GCN. In the second study, we create less powerful node features by taking only a randomly sampled subset of 30 features for clustering, training, and testing. In this case, having more node clusters reduces DP-GCN's accuracy. When the node features are less powerful, DP-GCN relies more on the graph connection information to make predictions and therefore having an adjacency matrix that is more robust to noise is more important than preserving fine-grained connection information.

8.3 CONCLUSION

Overall, in this chapter we propose the first differentially private GCN model (DP-GCN). In particular, we show that based on the proposed hierarchical node clustering, DP-GCN is able to guarantee edge differential privacy while preserving high data utility on different large scale graph datasets. A series of ablation studies are conducted, from which we draw several interesting conclusions. For instance, we observe that with more node clusters, the utility of DP-GCN is improved, which leads to further discussion on how to reduce the cluster size and create more informative clusters. Our theoretic analysis and empirical observations will shed lights on future research towards training privacy preserving GCNs.

CHAPTER 9: CONCLUSION

This thesis studied the privacy risk in modern machine learning systems. We proposed more pragmatic attacks and empirical measurement metrics to better understand the risk. Based on this understanding, we designed three levels of privacy protections tailored to the utility and privacy requirements of different applications. Specifically, this thesis made the following contributions to help improve the community's understanding on privacy risk in machine learning systems.

First, we studied the influence of different adversarial knowledge in privacy attacks on machine learning models. Traditionally, attacks on machine learning models are divided into two categories: white-box attacks and black-box attacks. In white-box attacks, the adversary has access to model parameters; and in black-box attacks, the adversary can only access the model's predictions on a limited number of queries. In this thesis, we analyzed the influence of other adversarial knowledge, including training data distribution, model structure, and the attack target. We demonstrated that the assumptions on the adversarial knowledge play an important role in evaluating privacy risks of machine learning models. As an example, we showed that, by targeting a few vulnerable records, it is possible to perform high-precision black-box membership inference attacks on well-generalized machine learning models, although these models are considered as having low-risk to black-box membership inference attacks in general.

Second, this thesis categorized privacy protections into three distinct levels based on the trade-off between utility and privacy. This categorization takes a novel approach to balance the privacy-utility trade-off along two dimensions (1) what privacy protection criteria should be applied and (2) whether the data is to be shared for general uses or specific applications.

Along the first dimension, we proposed test-based empirical obfuscations to provide privacy protections for cases where achieving strong theoretical privacy guarantee is challenging. We took system logs as an example under this category due to their high dimensionality and strong utility requirements. We designed a general framework to support sharing system logs while protecting user-specified sensitive attributes. The framework uses hypothesis tests to identify potential leakage of the sensitive attributes and applies empirical obfuscations until all the tests are passed. Although this approach does not provide strong theoretical guarantee, it allows users to identify and mitigate substantial privacy risks before sharing the data.

Along the second dimension, we proposed differentially private algorithms to support two use cases of data sharing: (1) sharing synthetic data for general uses and (2) sharing a

machine learning model for a specific application. To support sharing of synthetic data, we proposed a scalable differentially private data generative model that could generate better quality and higher dimensional synthetic data compared to prior work. To support sharing of differentially private models, we proposed the first differentially private graph convolutional network for graph analysis.

Moving forward, the privacy and utility trade-off remains an intriguing problem with many research opportunities. We conclude by discussing two open questions spurred by this work.

First, could we design algorithms to support different levels of privacy-utility balance in the same dataset? This thesis categorized privacy protections into three levels based on the privacy and utility requirements of a dataset. However, each dataset contains records from different individuals, and these individuals may have various privacy preferences. Providing distinct privacy protections for records in the same dataset is an open research problem. Research along this direction would meet the increasingly diverse privacy and utility requirements of modern machine learning applications.

Second, could we combine the test-based privacy protection with strong theoretical protections such as differential privacy? In this thesis, we designed test-based empirical obfuscations to protect data with high utility requirements. Can we extend this approach to data that require stronger privacy protections? For example, can we use tests to help determine the appropriate privacy budget in differential privacy? Answering this question could help us preserve better utility for applications where obtaining a tight theoretical privacy bound is challenging.

REFERENCES

[1] O. M. Parkhi, A. Vedaldi, A. Zisserman et al., "Deep face recognition." in *bmvc*, vol. 1, no. 3, 2015, p. 6.

[2] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3061–3070.

[3] M. de Bruijne, "Machine learning approaches in medical image analysis: From detection to diagnosis," 2016.

[4] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and structural biotechnology journal*, vol. 13, pp. 8–17, 2015.

[5] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 571–588, 2002.

[6] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[7] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[8] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2018, pp. 619–633.

[9] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Obfuscatory obscanturism: making workload traces of commercially-sensitive systems safe to release," in *2012 IEEE Network Operations and Management Symposium*. IEEE, 2012, pp. 1279–1286.

[10] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig, "Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays," *PLoS genetics*, vol. 4, no. 8, p. e1000167, 2008.

[11] E. A. Zerhouni and E. G. Nabel, "Protecting aggregate genomic data," *Science*, vol. 322, no. 5898, pp. 44–44, 2008.

[12] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: evaluating privacy leakage of generative models using generative adversarial networks," *arXiv preprint arXiv:1705.07663*, 2017.

[13] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

[14] G. Ateniese, G. Felici, L. V. Mancini, A. Spognardi, A. Villani, and D. Vitali, "Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers," *CoRR abs/1306.4447*, 2013.

[15] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing," in *Proceedings of the 2014 USENIX Security Symposium (USENIX Security)*. USENIX, 2014, pp. 17–32.

[16] M. Fredrikson, S. Jha, and T. Ristenpart, "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures," in *Proceedings of the 2015 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2015, pp. 1322–1333.

[17] F. Tramér, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs," in *Proceedings of the 2016 USENIX Security Symposium (USENIX Security)*. USENIX, 2016, pp. 601–618.

[18] B. Wang and N. Z. Gong, "Stealing Hyperparameters in Machine Learning," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018.

[19] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff Nets: Stealing Functionality of Black-Box Models," in *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019.

[20] D. Lowd and C. Meek, "Adversarial Learning," in *Proceedings of the 2005 ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2005, pp. 641–647.

[21] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *Proceedings of the 2015 International Conference on Learning Representations (ICLR)*, 2015.

[22] N. Papernot, P. D. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical Black-Box Attacks Against Machine Learning," in *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security (ASIACCS)*. ACM, 2017, pp. 506–519.

[23] B. Biggio, B. Nelson, and P. Laskov, "Poisoning Attacks against Support Vector Machines," in *Proceedings of the 2012 International Conference on Machine Learning (ICML)*. JMLR, 2012.

[24] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018.

[25] O. Suciu, R. Mărginean, Y. Kaya, H. D. III, and T. Dumitraş, "When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks," *CoRR abs/1803.06975*, 2018.

[26] A. Machanavajjhala, J. Gehrke, and M. Götz, "Data publishing against realistic adversaries," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 790–801, 2009.

[27] R. Bassily, A. Groce, J. Katz, and A. Smith, "Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 439–448.

[28] N. Li, W. Qardaji, and D. Su, "On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ACM, 2012, pp. 32–33.

[29] J. Lee and C. Clifton, "Differential identifiability," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 1041–1049.

[30] F. Tramèr, Z. Huang, J.-P. Hubaux, and E. Ayday, "Differential privacy with bounded priors: reconciling utility and privacy in genome-wide association studies," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1286–1297.

[31] N. Li, W. Qardaji, D. Su, Y. Wu, and W. Yang, "Membership privacy: a unifying framework for privacy definitions," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 889–900.

[32] E. Lui and R. Pass, "Outlier privacy," in *Theory of Cryptography*. Springer, 2015, pp. 277–305.

[33] J. Gehrke, M. Hay, E. Lui, and R. Pass, "Crowd-blending privacy," in *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 479–496.

[34] W. Qardaji, W. Yang, and N. Li, "Priview: practical differentially private release of marginal contingency tables," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 1435–1446.

[35] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Privbayes: Private data release via bayesian networks," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 4, p. 25, 2017.

[36] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," *arXiv preprint arXiv:1802.06739*, 2018.

[37] J. Yoon, J. Jordon, and M. van der Schaar, "PATE-GAN: Generating synthetic data with differential privacy guarantees," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=S1zk9iRqF7

[38] C. Dwork, A. Roth et al., "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[39] C. Rath, "Usable privacy-aware logging for unstructured log entries," in *Availability, Reliability and Security (ARES), 2016 11th International Conference on*. IEEE, 2016, pp. 272–277.

[40] J. Biskup and U. Flegel, "On pseudonymization of audit data for intrusion detection," in *Designing Privacy Enhancing Technologies*. Springer, 2001, pp. 161–180.

[41] J. Biskup and U. Flegel, "Transaction-based pseudonyms in audit data for privacy respecting intrusion detection," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2000, pp. 28–48.

[42] E. Lundin and E. Jonsson, "Privacy vs. intrusion detection analysis." in *Recent Advances in Intrusion Detection*, 1999.

[43] J. Xu, J. Fan, M. Ammar, and S. B. Moon, "On the design and performance of prefix-preserving ip traffic trace anonymization," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. ACM, 2001, pp. 263–266.

[44] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon, "Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme," in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. IEEE, 2002, pp. 280–289.

[45] J. Wilkes, "More Google cluster data," Google research blog, Nov. 2011, posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

[46] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov. 2011, revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data.

[47] B. Schneier and J. Kelsey, "Cryptographic support for secure logs on untrusted machines." in *USENIX Security Symposium*, vol. 98, 1998, pp. 53–62.

[48] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an encrypted and searchable audit log." in *NDSS*, vol. 4, 2004, pp. 5–6.

[49] K. Wouters, K. Simoens, D. Lathouwers, and B. Preneel, "Secure and privacy-friendly logging for egovernment services," in *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*. IEEE, 2008, pp. 1091–1096.

[50] T. Pulls, R. Peeters, and K. Wouters, "Distributed privacy-preserving transparency logging," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM, 2013, pp. 83–94.

[51] H. Nissenbaum and F. Brunton, "Vernacular resistance to data collection and analysis: A political theory," *First Monday*, vol. 16, no. 5, 2011.

[52] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1310–1321.

[53] D. E. Bakken, R. Rarameswaran, D. M. Blough, A. A. Franz, and T. J. Palmer, "Data obfuscation: Anonymity and desensitization of usable data sets," *IEEE Security & Privacy*, vol. 2, no. 6, pp. 34–41, 2004.

[54] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 2013, pp. 40–49.

[55] N. Bitansky and V. Vaikuntanathan, "Indistinguishability obfuscation from functional encryption," in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, 2015, pp. 171–190.

[56] X. He, A. Machanavajjhala, and B. Ding, "Blowfish privacy: Tuning privacy-utility trade-offs using policies," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 1447–1458.

[57] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *arXiv preprint arXiv:1802.04889*, 2018.

[58] M. Phillips and B. M. Knoppers, "The discombobulation of de-identification," *Nature biotechnology*, vol. 34, no. 11, p. 1102, 2016.

[59] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *IEEE Computer Security Foundations Symposium*, 2018.

[60] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," *arXiv preprint arXiv:1807.05852*, 2018.

[61] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 587–601.

[62] C. Dwork, "Differential privacy in the 40th international colloquium on automata," *Languages and Programming*, 2006.

[63] B. Efron, *The jackknife, the bootstrap and other resampling plans.* SIAM, 1982.

[64] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning.* ACM, 2009, pp. 609–616.

[65] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[66] N. Murata, S. Yoshizawa, and S.-i. Amari, "Network information criterion-determining the number of hidden units for an artificial neural network model," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 865–872, 1994.

[67] T. B. Sprague, "Shape preserving piecewise cubic interpolation," 1990.

[68] Y. Zhang, Z. L. Liu, and M. Song, "Chinet uncovers rewired transcription subnetworks in tolerant yeast for advanced biofuels conversion," *Nucleic acids research*, vol. 43, no. 9, pp. 4393–4407, 2015.

[69] C. Gentile and M. K. Warmuth, "Linear hinge loss and average margin," in *Advances in Neural Information Processing Systems*, 1999, pp. 225–231.

[70] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[71] R. Watrigant, M. Bougeret, and R. Giroudeau, "The k-sparsest subgraph problem," 2012.

[72] A. K. Jain and R. C. Dubes, *Algorithms for clustering data.* Prentice-Hall, Inc., 1988.

[73] J. T. Kost and M. P. McDermott, "Combining dependent p-values," *Statistics & Probability Letters*, vol. 60, no. 2, pp. 183–190, 2002.

[74] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[75] Y. Wu, J. Padhye, R. Chandra, V. Padmanabhan, and P. A. Chou, "The local mixing problem," in *Proc. Information Theory and Applications Workshop*, 2006.

[76] Y. LeCun, C. Cortes, and C. J. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[77] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al., "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[78] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, 2019. [Online]. Available: https://doi.org/10.1109/SP.2019.00065 pp. 739–753.

[79] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2019, pp. 259–274.

[80] A. Johnson and V. Shmatikov, "Privacy-preserving data exploration in genome-wide association studies," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1079–1087.

[81] Y. Long, V. Bindschaedler, and C. A. Gunter, "Towards measuring membership privacy," *arXiv preprint arXiv:1712.09136*, 2017.

[82] "Amazon machine learning - predictive analytics with AWS," https://aws.amazon.com/machine-learning/, 2016, accessed: 2016-05-18.

[83] "Prediction API - pattern matching and machine learning," https://cloud.google.com/prediction/, 2016, accessed: 2016-05-18.

[84] "Machine learning | Microsoft Azure," https://azure.microsoft.com/en-us/services/machine-learning/, 2016, accessed: 2016-05-18.

[85] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis." in *USENIX Security Symposium*, 2016, pp. 601–618.

[86] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 17–32.

[87] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[88] K. Fukunaga and D. M. Hummels, "Leave-one-out procedures for nonparametric error estimates," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, no. 4, pp. 421–423, 1989.

[89] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." in *KDD*, vol. 96. Citeseer, 1996, pp. 202–207.

[90] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[91] I. Rish, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22. IBM New York, 2001, pp. 41–46.

[92] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1996, pp. 310–318.

[93] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Machine Learning*, vol. 59, no. 1-2, pp. 161–205, 2005.

[94] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[95] H. Schütze, "Introduction to information retrieval," in *Proceedings of the international communication of association for computing machinery conference*, 2008, p. 260.

[96] D. Roth, "Learning in natural language," *Urbana*, vol. 51, p. 61801, 1999.

[97] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[98] W. Fan, H. Wang, P. S. Yu, and S. Ma, "Is random model better? on its accuracy and efficiency," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 51–58.

[99] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A practical differentially private random decision tree classifier," in *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*. IEEE, 2009, pp. 114–121.

[100] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[101] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 3–18.

[102] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models," in *Proceedings of the 2019 Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[103] S. J. Oh, M. Augustin, B. Schiele, and M. Fritz, "Towards Reverse-Engineering Black-Box Neural Networks," in *Proceedings of the 2018 International Conference on Learning Representations (ICLR)*, 2018.

[104] J. Buolamwini and T. Gebru, "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification," in *Proceedings of the 2018 Conference on Fairness, Accountability, and Transparency (FAT*)*, 2018, pp. 77–91.

[105] M. Kearns, S. Neel, A. Roth, and Z. S. Wu, "An Empirical Study of Rich Subgroup Fairness for Machine Learning," in *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency (FAT*)*. ACM, 2019, pp. 100–109.

[106] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in neural information processing systems*, 2017, pp. 3391–3401.

[107] https://archive.ics.uci.edu/ml/datasets/Census-Income+%28KDD%29.

[108] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[109] H. C. Assistance, "Summary of the hipaa privacy rule," *Office for Civil Rights*, 2003.

[110] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models," *CoRR abs/1806.01246*, 2018.

[111] B. C. Ross, "Mutual information between discrete and continuous data sets," *PloS one*, vol. 9, no. 2, p. e87357, 2014.

[112] *A Hypothesis Testing Approach to Sharing Logs with Confidence*. ACM, 2020.

[113] M. Xu, A. Papadimitriou, A. Feldman, and A. Haeberlen, "Using differential privacy to efficiently mitigate side channels in distributed analytics," in *Proceedings of the 11th European Workshop on Systems Security*. ACM, 2018, p. 4.

[114] Y. Yang, Z. Zhang, G. Miklau, M. Winslett, and X. Xiao, "Differential privacy in data publication and analysis," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 601–606.

[115] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 859–864.

[116] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "Towards automated log parsing for large-scale log data analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 931–944, 2018.

[117] "Apache spark," https://spark.apache.org/.

[118] C. E. McCulloch and J. M. Neuhaus, "Generalized linear mixed models," *Encyclopedia of biostatistics*, vol. 4, 2005.

[119] T. Kalibera, L. Bulej, and P. Tuma, "Benchmark precision and random initial state," in *Proceedings of the 2005 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2005)*, 2005, pp. 484–490.

[120] A. Maricq, D. Duplyakin, I. Jimenez, C. Maltzahn, R. Stutsman, and R. Ricci, "Taming performance variability," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 409–425.

[121] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Booz-Allen and Hamilton Inc Mclean Va, Tech. Rep., 2001.

[122] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012.

[123] N. Mantel, "Chi-square tests with one degree of freedom; extensions of the mantel-haenszel procedure," *Journal of the American Statistical Association*, vol. 58, no. 303, pp. 690–700, 1963.

[124] Z. Ding, Y. Wang, G. Wang, D. Zhang, and D. Kifer, "Detecting violations of differential privacy," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 475–489.

[125] Student, "The probable error of a mean," *Biometrika*, pp. 1–25, 1908.

[126] M. A. Stephens, "Edf statistics for goodness of fit and some comparisons," *Journal of the American statistical Association*, vol. 69, no. 347, pp. 730–737, 1974.

[127] M. Bland, "Do baseline p-values follow a uniform distribution in randomised trials?" *PloS one*, vol. 8, no. 10, p. e76010, 2013.

[128] R. A. Fisher, *Statistical methods for research workers*. Genesis Publishing Pvt Ltd, 2006.

[129] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 619–633.

[130] S. Sebastio, K. S. Trivedi, and J. Alonso, "Characterizing machines lifecycle in google data centers," *Performance Evaluation*, vol. 126, pp. 39 – 63, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016653161830004X

[131] K. Ousterhout, R. Rasti, S. Ratnasamy, S. Shenker, and B.-G. Chun, "Making sense of performance in data analytics frameworks," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, 2015, pp. 293–307.

[132] D. Lion, A. Chiu, H. Sun, X. Zhuang, N. Grcevski, and D. Yuan, "Don't get caught in the cold, warm-up your {JVM}: Understand and eliminate {JVM} warm-up overhead in data-parallel systems," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 383–400.

[133] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: efficient performance prediction for large-scale advanced analytics," in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, 2016, pp. 363–378.

[134] K. Ousterhout, C. Canel, S. Ratnasamy, and S. Shenker, "Monotasks: Architecting for performance clarity in data analytics frameworks," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 184–200.

[135] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, "Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 469–482.

[136] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 559–570, 2013.

[137] M. Chiappetta, E. Savas, and C. Yilmaz, "Real time detection of cache-based side-channel attacks using hardware performance counters," *Applied Soft Computing*, vol. 49, pp. 1162–1174, 2016.

[138] R. Tahir, M. Huzaifa, A. Das, M. Ahmad, C. Gunter, F. Zaffar, M. Caesar, and N. Borisov, "Mining on someone else's dime: Mitigating covert mining operations in clouds and enterprises," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2017, pp. 287–310.

[139] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019, pp. 1–19.

[140] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin et al., "Meltdown: Reading kernel memory from user space," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 973–990.

[141] Y. Long, S. Lin, Z. Yang, C. A. Gunter, and B. Li, "Scalable differentially private generative student model via pate," *arXiv preprint arXiv:1906.09338*, 2019.

[142] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[143] C. Dwork and V. Feldman, "Privacy-preserving prediction," *arXiv preprint arXiv:1803.10266*, 2018.

[144] I. Mironov, "Renyi differential privacy," in *Computer Security Foundations Symposium (CSF), 2017 IEEE 30th.* IEEE, 2017, pp. 263–275.

[145] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=HkwoSDPgg

[146] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and U. Erlingsson, "Scalable private learning with PATE," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rkZB1XbRZ

[147] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2001, pp. 245–250.

[148] S. Gong, V. N. Boddeti, and A. K. Jain, "On the intrinsic dimensionality of face representation," *CoRR*, vol. abs/1803.09672, 2018. [Online]. Available: http://arxiv.org/abs/1803.09672

[149] C. Xu, J. Ren, Y. Zhang, Z. Qin, and K. Ren, "Dppro: Differentially private high-dimensional data release via random projection," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 3081–3093, 2017.

[150] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in *2015 IEEE Symposium Series on Computational Intelligence.* IEEE, 2015, pp. 159–166.

[151] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[152] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[153] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[154] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[155] L. Liu, J. Wang, J. Liu, and J. Zhang, "Privacy preservation in social networks with sensitive edge weights," in *proceedings of the 2009 SIAM International Conference on Data Mining.* SIAM, 2009, pp. 954–965.

[156] L. Liu, J. Wang, J. Liu, and J. Zhang, "Privacy preserving in social networks against sensitive edge disclosure," Technical Report Technical Report CMIDA-HiPSCCS 006-08, Department of . . . , Tech. Rep., 2008.

[157] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18 209–18 237, 2018.

[158] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008. [Online]. Available: http://www.cs.iit.edu/~ml/pdfs/sen-aimag08.pdf

[159] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," 2019.

[160] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[161] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *CoRR*, vol. abs/1603.08861, 2016. [Online]. Available: http://arxiv.org/abs/1603.08861

[162] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*. OpenReview.net, 2017.

[163] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[164] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, March 1982.