Machine learning methods for privacy protection : leakage measurement and mechanisms design

Marco Romanelli

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2020IPPAX045

Thèse de doctorat

Méthodes d'apprentissage machine pour la protection de la vie privée: mesure de leakage et design des mécanismes

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à École polytechnique et Università di Siena

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Informatique, données et IA

Thèse présentée et soutenue à Palaiseau, le 21 Octobre 2020, par

M. MARCO ROMANELLI

Composition du Jury :

Haar Stefan Directeur de recherche, LSV, CNRS & ENS	Président
Gambs Sébastien Professeur, Univeristé du Québec à Montreal (UQAM))	Rapporteur
Malacaria Pasquale Professeur, Queen Mary University of London	Rapporteur
Véronique Cortier Directeur de recherche, Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA)	Examinateur
Erkip Elza Professeur, New York University	Examinateur
Piantanida Pablo Professeur, L2S at CentraleSupelec, CNRS, Université Paris Saclay	Examinateur
Palamidessi Catuscia Directeur de recherche, Inria, École Polytechnique & IPP	Directeur de thèse
Falaschi Moreno Professeur, Università di Siena	Co-directeur de thèse

Institute Polytechnique de Paris
Thése de Doctorat
Spécialité Informatique

Machine learning methods for privacy protection: leakage measurement and mechanism design

Marco Romanelli

Rapporteurs:	Sébastien GAMBS Pasquale MALACARIA
Directeur de thése:	Catuscia PALAMIDESSI
Co-Directeur de thése:	Konstantinos CHATZIKOKOLAKIS
Directeur de cotutelle:	Moreno FALASCHI
Examinateurs:	Véronique CORTIER Elza ERKIP Stefan HAAR (Président du Jury) Pablo PIANTANIDA

Abstract

In recent years, there has been an increasing involvement of artificial intelligence and machine learning (ML) in countless aspects of our daily lives. In this PhD thesis, we study how notions of information theory and ML can be used to better measure and understand the information leaked by data and / or models, and to design solutions to protect the privacy of the shared information.

We first explore the application of ML to estimate the information leakage of a system. We consider a black-box scenario where the system's internals are either unknown, or too complicated to analyze, and the only available information are pairs of input-output data samples. Previous works focused on counting the frequencies to estimate the input-output conditional probabilities (frequentist approach), however this method is not accurate when the domain of possible outputs is large. To overcome this difficulty, the estimation of the Bayes error of the ideal classifier was recently investigated using ML models and it has been shown to be more accurate thanks to the ability of those models to learn the input-output correspondence. However, the Bayes vulnerability is only suitable to describe one-try attacks. A more general and flexible measure of leakage is the g-vulnerability, which encompasses several different types of adversaries, with different goals and capabilities. We therefore propose a novel ML based approach, that relies on data preprocessing, to perform black-box estimation of the g-vulnerability, formally studying the learnability for all data distributions and evaluating performances in various experimental settings.

In the second part of this thesis, we address the problem of obfuscating sensitive information while preserving utility, and we propose a ML approach inspired by the generative adversarial networks paradigm. The idea is to set up two nets: the generator, that tries to produce an optimal obfuscation mechanism to protect the data, and the classifier, that tries to de-obfuscate the data. By letting the two nets compete against each other, the mechanism improves its degree of protection, until an equilibrium is reached. We apply our method to the case of location privacy, and we perform experiments on synthetic data and on real data from the Gowalla dataset. The performance of the obtained obfuscation mechanism is evaluated in terms of the Bayes error, which represents the strongest possible adversary.

Finally, we consider that, in classification problems, we try to predict classes observing the values of the features that represent the input samples. Classes and features' values can be considered respectively as secret input and observable outputs of a system. Therefore, measuring the leakage of such a system is a strategy to tell the most and least informative features apart. Information theory can be considered a useful concept for this task, as the

prediction power stems from the correlation, i.e., the mutual information, between features and labels. We compare the Shannon entropy based mutual information to the Rényi min-entropy based one, both from the theoretical and experimental point of view showing that, in general, the two approaches are incomparable, in the sense that, depending on the considered dataset, sometimes the Shannon entropy based method outperforms the Rényi min-entropy based one and sometimes the opposite occurs.

Résumé

Ces dernières années, l'intelligence artificielle et l'apprentissage machine (ML) ont été de plus en plus présents dans d'innombrables aspects de notre vie quotidienne. Dans cette thèse de doctorat, nous étudions comment les notions de théorie de l'information et de ML peuvent être utilisées pour mieux mesurer et comprendre les informations divulguées par les données et/ou les modèles, et pour concevoir des solutions visant à protéger la confidentialité des informations partagées.

Nous explorons d'abord l'application du ML pour estimer l'information leakage d'un système. Nous envisageons un scénario black-box dans lequel les éléments internes du système sont inconnus, ou trop compliqués à analyser, et les seules informations disponibles sont des paires de données input-output. Les travaux précédents se sont concentrés sur le comptage des fréquences pour estimer les probabilités conditionnelles d'input-output (frequentist approach), cependant cette méthode n'est pas précise lorsque le domaine des outputs possibles est large. Pour surmonter cette difficulté, l'estimation par ML de l'erreur du classificateur idéal (Bayes) a récemment été étudiée et sa précision supérieure, grâce à la capacité des modèles à apprendre la correspondance input-output, a été démontré. Cependant, la Bayes vulnerability ne convient que pour décrire des attaques one-try. Une mesure plus générale est la g-vulnerability, qui englobe plusieurs types d'adversaires, avec des objectifs et des capacités différents. Nous proposons donc une nouvelle approche basée sur la ML, qui repose sur le pre-processing des données, pour effectuer une estimation black-box de la g-vulnerability, en étudiant formellement la capacité d'apprentissage pour toutes les distributions de données et en évaluant les performances dans divers contextes expérimentaux.

Dans la deuxième partie de cette thèse, nous abordons le problème de l'obscurcissement des informations sensibles tout en préservant leur utilité, et nous proposons une approche de ML inspirée du paradigme generative adversarial nets. L'idée est de mettre en place deux réseaux : le générateur, qui essaie de produire un mécanisme d'obscurcissement optimal pour protéger les données, et le classificateur, qui essaie de désobstruer les données. En laissant les deux réseaux se concurrencer, le mécanisme améliore son degré de protection, jusqu'à ce qu'un équilibre soit atteint. Nous appliquons notre méthode au cas de la location privacy, et nous effectuons des expériences sur des données synthétiques et sur des données réelles provenant de le dataset Gowalla. La performance du mécanisme d'obfuscation obtenu est évaluée en fonction de l'erreur de Bayes, qui représente l'adversaire le plus fort possible.

Enfin, nous considérons que, dans les problèmes de classification, nous essayons de prévoir les classes en observant les valeurs des caractéristiques qui représentent les échantillons

d'entrée. Les valeurs des classes et des caractéristiques peuvent être considérées respectivement comme des inputs secrètes et des outputs observables d'un système. Par conséquent, la mesure de information leakage d'un tel système est une stratégie permettant de distinguer les caractéristiques les plus et les moins informatives. La théorie de l'information peut être considérée comme un concept utile pour cette tâche, car le pouvoir de prédiction découle de la corrélation, c'est-à-dire de l'information mutuelle, entre les features et les labels. Nous comparons l'information mutuelle basée sur l'entropie de Shannon à celle basée sur la min-entropy de Rényi, tant du point de vue théorique qu'expérimental, en montrant qu'en général, les deux approches sont incomparables, dans le sens où, selon l'ensemble de données considéré, parfois la méthode basée sur l'entropie de Shannon surpasse celle basée sur la min-entropie de Rényi et parfois le contraire se produit.

Acknowledgements

I would like to express my special appreciation and thanks to my supervisor, Catuscia Palamidessi, for encouraging my research and for allowing me to grow both as a researcher and a person. I would also like to thank my cotutelle supervisor Moreno Falaschi and my co-supervisor Konstantinos Chatzikokolakis. I am grateful to them all for encouraging me to explore challenging research topics which resulted in consequential contributions to the privacy and security community. I would like to extend a special thanks to Pablo Piantanida, for challenging me to push my work further as a co-author, and mentoring me as a researcher.

This work would not have been possible without the financial support of Inria (Institut National de Recherche en Informatique et en Automatique) and the Programme Vinci at the Université franco-italienne, which have all my gratitude.

I am also deeply thankful to the reporters of my thesis, Sébastien Gambs and Pasquale Malacaria, for taking the time to evaluate this work. Additionally, I would like to thank the members of the jury, Véronique Cortier, Elza Erkip, and Stefan Haar, for letting my defense be an enjoyable moment, and for their brilliant comments and suggestions.

I would also like to thank the Comète team members: Ruta Binkyte-Sudauskiene, Sayan Biswas, Ehab ElSalamouny, Federica Granese, Ganesh Del Grosso, Gangsoo Jung, Santiago Quintero, and Frank Valencia. They are all amazing colleagues and I am glad to call them friends.

I would like to thank Valentina Castiglioni, whom I not only consider my friend, but also an outstanding researcher because of her scientific contributions and intellectual honesty. Her advice over the years has been most valuable.

Among the other colleagues I cannot forget to mention: Michell Guzman, who was the first member of the team to welcome me, and I am grateful for our friendship; Dimitri Thomopulos, whose words of wisdom, and incurable optimism have encouraged me to look at the big picture, and not obsess over a single problem; Gabriele Iommazzo, and Matteo Manighetti for working alongside me, I appreciate their companionship; Renan Spencer Trindade, a good friend who has constantly encouraged me to see the bright side.

I would also like to thank Georg Pichler, whom I consider an outstanding researcher, for always reminding me of the excitement brought by learning new things, as well as Matteo Tiezzi, an old colleague of mine and good friend, whom I thank for his support throughout the years.

I would like to thank Maria Agustina Ronco, and also Jessica Gameiro and other members of the Inria human resources staff. Their complete dedication to the research team made

it possible for us to focus on our work.

A special thanks to my family, my parents and sisters, who have supported me from Italy; not a single day went by without them showing their thoughts were always with me. I am also grateful to Ana and Jaime Rodriguez for encouraging me to persevere and have hope.

Finally, I would also like to thank my best friend and amazing partner, Sarai Rodriguez. She has covered the distance between New York City and Paris many times to be together, and has never let a day go by without checking in on me, encouraging me to value the most important things in life and supporting me when I was going through hard times. To her, I dedicate my work and efforts in the hope that we will soon be together.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

Nowadays, in the so called big data era, information represents one of the most valuable assets. Improvements in data storage resources, advancements in communication and information sharing infrastructures, together with the development of increasingly powerful hardware are the three pillars that have turned our society into a data-driven one.

The growth in the science and technology related to information storage has been exponential since mankind's first steps into this field. Let us consider that, while it took nearly 200 years from the first prototype of punching cards to their full adoption as the backbone of the American industrial and governmental communications in the 1950s, within the next forty years, both magnetic (tapes, floppy discs, etc.) and optical (compact disks, etc.) memory supports begun to take turns on the market. In the years 2000s flash drives have represented the best compromise between storage capabilities and compact dimensions. Today, cloud storage services are gaining more and more ground, providing three main benefits: accessibility from anywhere, low risk of system failure, and the possibility to use and deploy online services.

Moreover, the development of the internet, and especially the rise of the internet of things (IoT), has been contributing to the birth of the so called "smart world" which is consistently data-centered and relies on fast and stable connection infrastructures. Consequently, many online services have been growing at a fast pace in the last decade. For instance, GitHub, the famous repository-based cloud storage service, registered 10 million new users in 2019 alone[1]. Apple Music, one of the most popular music streaming services, has gone from 6.5 millions subscriptions in Q4 2015 to 68 millions in Q4 2019[2]. Netflix, the worldwide renowned movie and TV series streaming service, has grown from 21.6 millions subscribers in Q1 2012 to more than 180 millions in Q1 2020[3]. These are only some of the many possible

[1] According to a report on "The state of the octaverse", GitHub official blog.

[2] According to a worldwide survey published in April 2020 available at https://www.statista.com/statistics/604959/number-of-apple-music-subscribers/.

[3] According to a worldwide survey published in April 2020 available at https://www.statista.com/statistics/250934/quarterly-number-of-netflix-streaming-subscribers-worldwide/.

examples, but they are up to the point in showing how the development of the internet infrastructures has made it possible for services in the cloud to thrive. The new challenge for the years to come is the large scale deployment of 5G connectivity infrastructures that will allow us to move large amounts of data even faster than what we currently can[4].

Last but not least, the increasing power, availability and affordability of hardware dedicated to computation tasks (CPUs, GPUs, FPGAs, etc.) have been fundamental to push forward groundbreaking fields like machine learning (ML). Back in 1965, when Moore published his forecast on the yearly increase of number of transitors in the integrated circuits in [Moo65], it was clear that new discoveries would improve the level of technology and, in turn, this would make computational resources cheaper and sparkle new research as well.

However, if on the one hand relying on a wide variety of online services is evidently a benefit, on the other hand, threats to the privacy of users and their data have gradually become an uncomfortable reality that customers and service providers must face more and more frequently. In fact, once personal data is shared, it is no more exclusive property of the users and service providers must be trusted with handling potentially sensitive information. Therefore, companies that offer cloud services must provide guarantees such as the fact that data will not be sold without the owner's consent and also protection against malicious hacks. Lately, malicious attacks to privacy have become quite frequent and some has lead to huge scandals. Just to mention one, let us remind the case related to Facebook's privacy protection failure that in 2018 allowed Cambridge Analytica, a political consulting firm, to harvest data from millions of Facebook users' accounts data without consent. Prior to that, Facebook's weakness in preserving users' privacy had already been exposed by researchers at MIT who, back in 2005, managed to crawl and download personal data from several profiles even before the company had decided to allow official search engines to surf through their accounts. On top of this, similar scandals have involved many other big tech companies all around the world: Linked In in 2012, Yahoo in 2013, and Sina Weibo in 2020 just to mention a few.

Computer scientists have been focusing on exposing threats to privacy and weaknesses in privacy protection mechanisms. Attacks can be divided into two main categories: attacks against users' identities and attacks against users' data. The former aim at unveiling the users' real identities by linking the data related to an individual to real profiles. For instance, de-anonymization attacks fall in this category and one example is described in [NS08] where a de-anonymization algorithm is applied in order to link anonymous records (movie ratings information of 500,000 individuals contained in the Netflix Prize Dataset) to known Netflix users, using the information provided by IMDB.com. Attacks against users' data aim at increasing the attackers' accuracy when guessing real data from the modified version that is released by some protection mechanism. One example, related to location privacy, is retrieving the real location of a user from an obfuscated reported one.

[4]Experts estimate that the 5G connection will allow a top speed of approximately 10 Gbps, against the current state-of-the-art represented by LTE which tops at 300 Mbit/s (cfr. https://www.telekom.com/en/company/details/5g-speed-is-data-transmission-in-real-time-544498).

Lately, the application of ML to many fields has exacerbated the problem by proving that very powerful attacks can be modeled, especially via deep learning (cfr. [PSB+18]). Moreover, due to the fact that ML models are the engines under the hood of many cloud services, new attacks are possible. For instance, reconstruction attacks [GGH12] aim to retrieve the original data from the information which has been memorized in trained models. Model inversion attacks [FJR15] aim at collecting knowledge about the original data by observing the way trained models respond to generated samples. Membership inference attacks [SSSS17, PTC18, HMDC19, MSCS19] aim to determine whether a certain sample was in the training set used to train some observable models.

More recently, a new research direction has been gaining ground, i.e. the study of how notions of information theory and ML can be used to better measure and understand the information leaked by data and / or models, and to help design solutions to protect the privacy of the shared information. This thesis represents a contribution along this line of research. In particular, we focus on three aspects: how we can use ML to estimate the information leakage of a system, how we can use information theory notions to design "smart" privacy protection mechanisms and, finally, how measuring leakage can inform us about which features of the system are more informative from the point of view of the learning.

1.2 Leakage estimation

The information leakage of a system is a fundamental concern of computer security, and measuring the amount of sensitive information that an adversary can obtain by observing the outputs of a given system is of the utmost importance to understand whether such leakage can be tolerated or should be considered a major security flaw.

One important aspect to keep in mind when measuring leakage is the kind of attack that we want to model. In the seminal paper [KB07], the authors identified various kinds of adversaries and showed that they can be captured by known entropy measures. For instance, the Shannon entropy represents the expected number of binary queries that the adversary must submit to the system in order to fully determine the value of the secret.

In [Smi09], the Rényi min-entropy has been proposed to measure the system's leakage when the attacker has only one try at its disposal and attempts to make its best guess. The Rényi min-entropy is the logarithm of the Bayes vulnerability, which is the expected probability of success of the adversary that has exactly one attempt at his disposal (one-try), and tries to maximize the chance of guessing the right value of the secret. The Bayes vulnerability is the converse of the Bayes error, which was already proposed as a measure of leakage in [CPP08b].

A more general leakage measure should encompass many different adversaries, describing the attackers by means of a parametric function. An attempt to do so is represented by the work in [ACPS12], where the authors generalized the notion of Bayes vulnerability to that of g-vulnerability, by introducing a parametric term, i.e. the gain function g, that describes the adversary's payoff. The g-vulnerability is the expected gain of the adversary in a one-try attack. As we will explain in section 2.1, the notion of g-vulnerability implies a

notion of leakage, the g-leakage, which is directly derived from it (provided some a priori knowledge).

Much research effort has been dedicated to studying and proposing solutions to the problem of estimating the information leakage of a system, see for instance the works [CHM01, KB07, CPP08a, Bor09, Smi09, ACPS12, ACM+14, CFP19], just to mention a few. So far, this area of research, known as quantitative information flow (QIF), has mainly focused on the so-called white-box scenario. Namely, all these works assume that the system's channel, i.e. the conditional probabilities of the outputs (observables) given the inputs (secrets), is known, or can be computed by analyzing the system's internals. However, the white-box assumption is not always realistic. In fact, sometimes the system is unknown, or anyway it is too complex, so that an analytic computation becomes hard if not impossible to perform. Therefore, it is important to consider also a black-box approach where we only assume the availability of a finite set of input-output pairs generated by the system, possibly obtained by submitting queries or provided by a third party.

The estimation of the internal probabilities of a system's channel have been investigated in [CG11] and [CKN14] via a frequentist paradigm, i.e. relying on the computation of the frequencies of the outputs given some inputs. However, this approach does not scale to applications for which the output space is very large since a prohibitively large number of samples would be necessary to achieve good results and fails on continuous alphabets unless some strong assumption on the distributions are made. In order to overcome this limitation, the authors of [CCP19] exploited the fact that ML algorithms provide a better scalability to black-box measurements. Intuitively, the advantage of the ML approach over the frequentist one is its generalization power: while the frequentist method can only draw conclusions based on counts on the available samples, ML is able to extrapolate from the samples and provide better prediction (generalization) for the rest of the universe.

In the context of ML classification problems the classes can be considered secrets and the observables are represented by features (descriptors). Measuring a system's leakage can be useful to understand which features, among those which can be observed by an adversary, reveal more information about the classes. We could use this knowledge to select which features a data obfuscation mechanism should mainly focus on, to avoid adding noise on descriptors which already do not reveal much about the secrets. Otherwise this could also be used to understand which features, within a set of sensitive ones, mostly influences an unfair decision when those decisions are delegated to some automatic algorithm (cfr. [ABG+19]). A problem of different nature, which can be addressed and solved in a similar way, is that of data dimensionality reduction in classification problems. Among others approaches, filter methods, which reduce the amount of features by choosing a subset of the initially available ones, have been exploited in [Bat94, YM99, Fle04, PLD05, BHS15, BPZL12, VE14], just to mention some of the most important contributions on the topic. A straightforward way to measure the leakage of information between the classes and a set of features is computing their mutual information, which can be defined trough the notion of entropy already introduced at the beginning of this section. The idea is that the smaller is the conditional (aka residual) entropy of the classes given a certain set of features, the more likely the classi-

fication of a sample is to be correct. In principles, the set of features that maximizes the mutual information with the secrets is also the set on which one should focus when building a system for privacy protection.

1.3 Privacy mechanisms design

In the previous section, we have mentioned that measuring how much information is leaked by a given system represents a very important step to asses whether we can tolerate it or whether we should upgrade the current system to a more secure one. As we have seen so far, information theory provides a wide set of metrics to measure leakage with respect to different attack scenarios. Indeed, it is possible to model several adversaries and estimate how much information each one of them can extract from the output released by the system. We claim that this knowledge can be used to build and test privacy protection mechanisms so to come up with solutions that can reduce the leakage against various threats. Many research contributions have focused on mechanisms that create and release an obfuscated version of the original data, which reduces the leakage with respect to the sensitive information that we wish to conceal. Although, it is fundamental for the privacy mechanisms not to completely destroy the information conveyed by the original data, in order to maintain a reasonably good quality of the provided service (QoS). In other words, the utility loss of the designed obfuscation mechanisms should remain below a given threshold.

One straightforward way to build a mechanism that maximizes privacy and maintains a certain amount of utility is that of modeling the former as an objective function and the latter as a set of constraints. If objective function and constraints are linear this can be formulated in terms of linear programming. The works in [STT+12a, BCP14, OTPG17, STT17] are based on this idea. On the one hand, if feasible solutions are available and enough computational resources are available, linear programming, provides the optimal mechanism(s). On the other hand, the number of involved variables is often huge and the size of the corresponding linear programs limits the scalability of these methods to real world applications.

In section 1.2, we have also mentioned that ML can provide better models than those based on the system's internal probabilities empirical estimation. Intuitively, ML models can also be used to build privacy protection mechanisms. Recently, a new line of research has been proposing solutions based on the efficient optimization process of neural networks (the gradient descent) to retrieve a model that minimizes a loss function which encompasses both privacy and utility requirements. Among other works, [AA16,Ham17,HKC+17, TWI17,RLR18] rely on the notion of adversarial networks competing in a mini-max game to build privacy-protection mechanisms. In [JG18] an adversarial network framework is used to design privacy a protection mechanism against attribute inference attacks. The authors of [HO18] consider multi-party machine learning, and use adversarial training to mitigate privacy-related attacks such as party membership inference of individual records. The authors of [ES16] propose the minimax technique to remove private information from personal images. using a stochastic gradient alternate min-max optimizer. The authors of [RLR18]

consider personal images, and in particular the problem of preventing their re-identification while preserving their utility.

Other popular privacy obfuscation mechanisms are the results of studies concerning topics such as differential privacy (DP) [DMNS06], local differential privacy (LDP) [DJW13], and d-privacy [CABP13]. Although some of these mechanisms have been proved to be optimal against a specific kind of queries (cfr. geometric mechanism and counting queries), their main characteristic is that they are based on worst-case measures, ant therefore, they are concerned with the protection of each individual datum, which, in general, makes it harder to directly control the privacy-utility trade-off.

1.4 Goals

In this work, we are interested in investigating the application of information theory and ML notions in order to tackle problems related to the fields of privacy and security. In particular, we are interested in estimating the amount of information leaked by a given system, and designing a protection mechanism which, by means of controlled noise injection, protects sensitive data releasing an obfuscated version while maintaining the trade-off between the obtained privacy and the required utility.

To address the first point, and inspired by the novelties introduced by [CCP19], we aim at extending the leakage estimation to a framework that is concerned with a more general definition of adversary than just the Bayesian one. In order to do so, we aim at finding a way to estimate the g-leakage of a system and reducing this problem to that of approximating the error of the Bayes classifier without estimating the channel's internal conditional probabilities.

As to the second problem, inspired by previous work concerned with the use of adversarial networks in privacy preserving applications, we aim at building a framework in which a generative network enforces privacy protection via obfuscation by releasing a new version of the original data according to a new convenient distribution while also taking care of respecting the utility constraint imposed during the learning phase.

Finally, we consider the setting of a typical classification problem framework where the knowledge about classes comes from observing features. We focus on the use of the mutual information as a means to sort the features from the one that leaks the most information about the right classification down to the least informative one. We aim at evaluating the performances of a greedy algorithm that provides the aforementioned sort and is based once on the notion of Shannon entropy and once on the notion of Rényi min-entropy, comparing the outputs when the two different notions are applied. If on the one side we consider the framework of the dimensionality reduction problem in ML, on the other side one can exploit such an algorithm to see which features require more attention from the privacy defense standpoint, i.e. which of them leak the most about the classes.

1.5 Plan of the thesis and contributions

In this section we present a brief description of the content of the following chapters, highlighting the main contributions in each of them.

In chapter 2 we provide a review of the information theory and QIF notions that will be used later on. In particular, we focus on the general definitions, leaving the specific case related details to later on when they will be needed. We briefly recall notions about ML as well.

In chapter 3 we propose a ML learning based solution for estimating the information leakage of a system. Given its flexibility and capacity to encompass many different scenarios of attacks against privacy, we consider the problem of estimating the g-vulnerability of a system. We provide statistical guarantees showing the learnability of the g-vulnerability for all distributions and we derive distribution-free bounds on the accuracy of its estimation. We introduce two pre-processing methods that we use in the process of reducing the problem to that of approximating the Bayes classifier, so that any universally consistent ML algorithm can be used for the purpose. This reduction essentially takes into account the impact of the adversary's gain function in the generation of the training data. The last part of this chapter shows several practical applications of the proposed estimation framework to different privacy attacks scenario.

Chapter 4 is concerned with the design of a machine learning based privacy protection mechanism. We propose a general approach based on adversarial nets to generate obfuscation mechanisms with a good privacy utility trade-off that can be directly controlled at training time. The underlining idea is that reducing the mutual information between the data released by the protection mechanism and the corresponding labels would reduce the leakage of what an adversary can observe. For the experimental part, we focus on the location privacy scenario, providing results for application to both synthetic and real data when different utility constraints are required. We evaluate the privacy of the mechanism in terms of the Bayes error, which represents the strongest possible adversary, and we compare the privacy-utility trade-off of our method with that of the planar Laplace mechanism and the optimal linear programming based solution (where possible).

In chapter 5, we present a framework to detect the most meaningful features in classification problems where the aim is to predict classes observing the values of the features that represent the input samples. Classes and features' values can be considered respectively as secret input and observable outputs of a system. Therefore, measuring the leakage of such a system is a strategy to tell the most and least informative features apart. Information theory can be considered a useful concept for this task, as the prediction power stems from the correlation, i.e., the mutual information, between features and labels. Many algorithms for feature selection in the literature have adopted the Shannon entropy based mutual information. We explore the possibility of using Rényi min-entropy instead, given its strict relation to the notion of Bayes error. We prove that in general the two approaches are incomparable, in the sense that we show that it is possible to build datasets on which the Rényi-based algorithm performs better than the corresponding Shannon-based one, and datasets on which

the opposite occurs. We run experiments on three benchmark datasets and we observe that, by selecting the most informative features through the Rényi min-entropy, we achieve better classification performances.

Finally, in chapter 6, we present the concluding remarks of this work.

Publications from this dissertation

The content of this dissertation is based on the following publications:

- Chapter 3 is based on the results presented in **Estimating g-leakage via machine learning** [RCPP20], that will appear in the proceedings of the 27th *ACM SIGSAC Conference on Computer and Communications Security (CCS 2020)*.
- Chapter 4 is based on the results presented in **Generating optimal privacy-protection mechanisms via machine learning** [RCP20], that appeared in the proceedings of the 33rd *IEEE Computer Security Foundations Symposium (CSF 2020)*. A short version of this work has been presented at the 2nd *Privacy Preserving Machine Learning Workshop (PPML 2019)* co-located with 26th *ACM SIGSAC Conference on Computer and Communications Security (CCS 2019)*.
- Chapter 5 is based on the results presented in **Feature selection with Rényi min-entropy** [PR18], that appeared in the proceedings of the 8th *International Association for Pattern Recognition TC3 Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR 2018)*.

Other publications

Other works I have contributed to during my :

- **Modern Applications of Game-Theoretic Principles** (Invited Paper), that appeared in the proceedings of the 31st *International Conference on Concurrency Theory (CONCUR 2020)*.
- **Derivation of Constraints from Machine Learning Models and Applications to Security and Privacy** that will appear in the proceedings of *Recent Developments of the Design and Implementation of Programming Languages 2020 (DIP 2020)*.

CHAPTER 2

Preliminaries

In this chapter, we review some useful notions from QIF, such as g-leakage and g-vulnerability. We then move on to recalling classical information theory notions such as Shannon entropy, Rényi min-entropy, Bayes risk and mutual information. In doing so, we also introduce some notation that will be used throughout the rest of this work, reserving the right to add specific notation details on a case-by-case basis, where needed, in the following chapters. At the end of this chapter we briefly discuss some ML related notions, in particular artificial neural networks (ANN) and k-nearest neighbors (k-NN) algorithm.

2.1 Quantitative information flow notions: g-leakage and g-vulnerability

Let $\mathcal{X}$ be a set of secrets and $\mathcal{Y}$ a set of observations. The adversary's initial knowledge about the secrets is modeled by a prior distribution $\mathcal{P}(\mathcal{X})$ (namely P_X, and often referred to as π). A system is modeled as a probabilistic *channel* from $\mathcal{X}$ to $\mathcal{Y}$, described by a stochastic matrix C, whose elements C_{xy} give the probability to observe $y \in \mathcal{Y}$ when the input is $x \in \mathcal{X}$ (namely $P_{Y|X}$). Running C with input π induces a joint distribution on $\mathcal{X} \times \mathcal{Y}$ denoted by $\pi \triangleright C$.

In the g-leakage framework [ACPS12] an adversary is described by a set $\mathcal{W}$ of *guesses* (or *actions*) that it can make about the secret, and by a *gain function* $g(w, x)$ expressing the gain of selecting the guess w when the real secret is x. The prior g-vulnerability is the *expected gain* of an optimal guess, given a prior distribution on secrets:

$$V_g(\pi) \stackrel{\text{def}}{=} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi_x \cdot g(w, x)\,. \tag{2.1}$$

In the posterior case, the adversary observes the output of the system which allows to improve its guess. Its expected gain is given by the posterior g-vulnerability, according to

$$V_g(\pi, C) \stackrel{\text{def}}{=} \sum_{y \in \mathcal{Y}} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi_x \cdot C_{xy} \cdot g(w, x)\,. \tag{2.2}$$

Finally, the multiplicative[1] and additive g-leakage quantify how much a specific channel C *increases* the vulnerability of the system:

$$\mathcal{L}_g^{\mathrm{M}}(\pi, C) \stackrel{\text{def}}{=} \frac{V_g(\pi, C)}{V_g(\pi)}\,, \quad \mathcal{L}_g^{\mathrm{A}}(\pi, C) \stackrel{\text{def}}{=} V_g(\pi, C) - V_g(\pi)\,. \tag{2.3}$$

The choice of the gain function g allows to model a variety of different adversarial scenarios. The simplest case is the *identity* gain function, defined as

$$g_{\mathrm{id}}(w, x) = \begin{cases} 1, & \text{iff } x = w \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

and given by $\mathcal{W} = \mathcal{X}$. This gain function models an adversary that tries to guess the secret exactly in one try; $V_{g_{\mathrm{id}}}$ is the *Bayes-vulnerability*, which corresponds to the complement of the Bayes error (cfr. [ACPS12]).

However, the interest in the g-vulnerability lies on the fact that many more adversarial scenarios can be captured by a proper choice of g. For instance, taking $\mathcal{W} = \mathcal{X}^k$ with

$$g_{\mathrm{id}}(w, x) = \begin{cases} 1, & \text{iff } x \in w \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

models an adversary that tries to guess the secret correctly in k tries. Moreover, guessing the secret *approximately* can be easily expressed by constructing g from a metric d on $\mathcal{X}$; this is a standard approach in the area of *location privacy* [STBH11, STT+12b] where $g(w, x)$ is taken to be inversely proportional to the Euclidean distance between w and x. Several other gain functions are discussed in [ACPS12], while [ACM+16] shows that *any* vulnerability function satisfying basic axioms can be expressed as V_g for a properly constructed g.

Note that, given $V_g(\pi, C)$, estimating $\mathcal{L}_g^{\mathrm{M}}(\pi, C)$ and $\mathcal{L}_g^{\mathrm{A}}(\pi, C)$ is straightforward, since $V_g(\pi)$ only depends on the prior (not on the system) and it can be either computed analytically or estimated from the samples.

2.2 Information theory notions

We are going to we briefly cover some basic notions from the fields of information and probability theory. i.e. entropy and mutual information. We refer to [CT91] for further details.

Let X, Y be discrete random variables with respectively n and m possible values: $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$. Let $p_X(\cdot)$ and $p_Y(\cdot)$ indicate the probability distribution associated to X and Y respectively, and let $p_{Y,X}(\cdot, \cdot)$ and $p_{Y|X}(\cdot|\cdot)$ indicate the joint

[1] In the original paper, the multiplicative version of g-leakage was defined as the $\log$ of the definition given here. In recent literature, however, the $\log$ is not used anymore. Anyway, the two definitions are equivalent from the point of view of comparing systems, since $\log$ is a monotonic function.

and the conditional probability distributions, respectively. Namely, $p_{Y,X}(x, y)$ represents the probability that $X = x$ and $Y = y$, while $p_{Y|X}(y|x)$ represents the probability that $Y = y$ given that $X = x$. For simplicity, when clear from the context, we will omit the subscript, and write for instance $p(x)$ instead of $p_X(x)$. Conditional and joint probabilities are related by the chain rule

$$p(x, y) = p(x)\, p(y|x),$$

from which (by the commutativity of $p(x, y)$) we can derive the Bayes theorem:

$$p(x|y) = \frac{p(y|x)\, p(x)}{p(y)}.$$

The Rényi entropies ([Rén61]) are a family of functions representing the uncertainty associated to a random variable. Each Rényi entropy is characterized by a non-negative real number α (order), with $\alpha \neq 1$, and is defined as

$$H_\alpha(X) \stackrel{\text{def}}{=} \frac{1}{1-\alpha} \log \left(\sum_{i=1}^{n} p(x_i)^\alpha \right). \tag{2.6}$$

If $p(\cdot)$ is uniform then all the Rényi entropies are equal to $\log |X|$. Otherwise they are weakly decreasing in α. Shannon and min-entropy are particular cases:

$$\begin{array}{ll} \alpha \to 1 & H_1(X) = -\sum_x p(x) \log p(x) \quad \text{Shannon entropy,} \\ \alpha \to \infty & H_\infty(X) = -\log \max_x p(x) \quad \text{min-entropy.} \end{array}$$

Let $H_1(X, Y)$ represent the joint entropy X and Y. Shannon *conditional entropy* of X given Y is the average residual entropy of X once Y is known, and it is defined as

$$H_1(X|Y) \stackrel{\text{def}}{=} -\sum_{xy} p(x, y) \log p(x|y) = H_1(X, Y) - H_1(Y). \tag{2.7}$$

Shannon *mutual information* of X and Y represents the correlation of information between X and Y, and it is defined as

$$I_1(X; Y) \stackrel{\text{def}}{=} H_1(X) - H_1(X|Y) = H_1(X) + H_1(Y) - H_1(X, Y). \tag{2.8}$$

It is possible to show that $I_1(X; Y) \geqslant 0$, with $I_1(X; Y) = 0$ iff X and Y are independent, and that $I_1(X; Y) = I_1(Y; X)$. Finally, *Shannon conditional mutual information* is defined as:

$$I_1(X; Y|Z) \stackrel{\text{def}}{=} H_1(X|Z) - H_1(X|Y, Z), \tag{2.9}$$

In the following chapters, we will use H and I instead of H_1 and I_1 when the context does not require for order $\alpha = 1$ to be specified.

Recently, some advances in the fields of security and privacy have revived the interest for the *Rényi min-entropy*. However, Rényi did not define the conditional min-entropy. Therefore, there have been various proposals, in particular those in [Ari75], [Sib69], and [Csi95].

Moreover, [Cac97] defined the conditional min-entropy of X given Y along the lines of conditional Shannon entropy, namely as the expected value of the entropy of X for each given value of Y. Such definition, however, violates the *monotonicity property*. Namely, knowing the value of Y could increase the entropy of X rather than diminishing it.

We use the version of [Smi09]:

$$H_\infty(X|Y) \stackrel{\text{def}}{=} -\log \sum_y \max_x (p(y|x)p(x)). \tag{2.10}$$

This definition closely corresponds to the Bayes risk, i.e., the expected error when we try to guess X once we know Y, formally defined as

$$\mathcal{B}(X\,|\,Y) \stackrel{\text{def}}{=} 1 - \sum_y p(y) \max_x p(x|y). \tag{2.11}$$

The reason behind the importance of this notion is that it models a basic notion of attacker: the (*one-try*) *eavesdropper*. Such attacker tries to infer a secret (e.g., a key, a password, etc.) from the observable behavior of the system trying to minimize the probability of error.

The *Rényi mutual information* is defined as:

$$I_\infty(X;Y) \stackrel{\text{def}}{=} H_\infty(X) - H_\infty(X|Y). \tag{2.12}$$

It is possible to show that $I_\infty(X;Y) \geqslant 0$, and that $I_\infty(X;Y) = 0$ if X and Y are independent (the reverse is not necessarily true). Contrary to Shannon mutual information, I_∞ is not symmetric. The conditional mutual information is defined as

$$I_\infty(X;Y|Z) \stackrel{\text{def}}{=} H_\infty(X|Z) - H_\infty(X|Y,Z). \tag{2.13}$$

2.3 Brief review of machine learning notions

General notions of learning theory

We give here a brief introduction about the learning process and the derivation of the model, and we will focus on the supervised learning scenario in the context of classification problems. We describe the basic elements common to all learning algorithms, and to this purpose we introduce a generic learner model based on a well established statistic framework.

A learning problem is defined by:

- a domain $\mathcal{X}$ of objects, represented as a vector of *features* (aka *attributes*) that we would like to classify;
- a set of *labels* (aka *classes*) $\mathcal{Y}$;
- a set of training data, i.e., a sequence $\mathcal{S} = ((\vec{x}_1; y_1) \ldots (\vec{x}_m; y_m))$ of pairs in $\mathcal{X} \times \mathcal{Y}$;

- a correct labelling function $f : \mathcal{X} \to \mathcal{Y}$, such that, for all i, $y_i = f(\vec{x}_i)$;
- a distribution $\mathbb{D}\mathcal{X}$, according to which the samples are generated;
- the *prediction rule* or *hypothesis* $h : \mathcal{X} \to \mathcal{Y}$, that can be used to predict the label of new domain points;
- a measure of success that quantifies the predictor's error.

Ideally, the goal of the learning process is to select an h that minimizes the *risk*, defined as:

$$L_{\mathbb{D},f}(h) \stackrel{\text{def}}{=} \mathbb{P}_{\vec{x}\sim\mathbb{D}}\left[h(\vec{x}) \neq f(\vec{x})\right], \tag{2.14}$$

which represents the probability ($\mathbb{P}$) of a mismatch between h and f, measured with respect to the distribution $\mathbb{D}$. The quantity in eq. (2.14) is the error of a specific prediction rule h, or as referred at in many context, it is its *generalization error*. This error can be computed on the training set (training error) or on a test set of samples drawn from the same distribution as the training set but never seen during the training phase (test error). If the error on the training data is large, then this usually means that the model is "underfitting" the data distribution. However, a negligible training error does not necessarily mean that our model is good. Indeed, if the error on the test set is large, it is likely that the trained model overfits the training data.

In practice, however, we cannot compute analytically the h that minimizes (2.14), because we do not have a mathematical description of $\mathbb{D}$. What we can do, instead, is to use the training set $\mathcal{S}$, that, being generated from $\mathbb{D}$, represents an approximation of it. Then h is selected so to minimize the *empirical risk* over m samples, which is defined as:

$$L_{\mathcal{S}}(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(\vec{x}_i) \neq y_i\}|}{m}. \tag{2.15}$$

This principle is called *empirical risk minimization* (ERM). The way this minimization is achieved depends on the specific algorithm, and the function h that is derived is called *model*.

For an extended discussion of the topic as well as a more complete overview of the learning problem we refer to [SSBD14, Val84]. For further information about ML and the most popular algorithms and applications we refer to [GBC16], while for a more theoretical and statistical background on the learning problem we refer to [DGL96, HTF09].

Artificial neural networks

We provide a short review of the aspects of ANN that are relevant for this work. Since a thorough discussion would be beyond the scope of this work, we refer to ([Bis07, GBC16, HTF01] for an in-depth study. Neural networks represent an attempt to reproduce the behavior of the brain's cells and are usually modeled as directed graphs with weights on the connections and nodes that forward information through "activation functions", often introducing non-linearity (such as sigmoids or soft-max). In particular, we consider an instance

of learning known as *supervised learning*, where input samples are provided to the network model together with target labels (supervision). From the provided data and by means of iterative updates of the connection weights, the network learns how the data and respective labels are distributed. The training procedure, known as back-propagation, is an optimization problem aimed at minimizing a loss function that quantifies the quality of the network's prediction with respect to the data.

Classification problems are a typical example of tasks for which supervised learning works well. Samples are provided together with target labels which represent the classes they belong to. A model can be trained using these samples and, later on, it can be used to predict the class of new samples.

The No Free Lunch theorem (NFL) [Wol96] holds for ANN as it does for all the ML approaches. Therefore, in general, it cannot be said that ANN are better than other ML methods. However, it is well known that the NFL can be broken by additional information on the data or the particular problem we want to tackle, and, nowadays, for most applications and available data, especially in multidimensional domains, ANN models outperform other methods and therefore they represent the state of the art.

k-Nearest Neighbors

The k-NN algorithm is one of the simplest algorithms used to classify a new sample given a training set of samples labelled as belonging to specific classes. This algorithm assumes that the space of the features is equipped with a notion of distance. The basic idea is the following: every time we need to classify a new sample, we find the k samples whose features are closest to those of the new one (nearest neighbors). Once the k nearest neighbors are selected, a majority vote over their class labels is performed to decide which class should be assigned to the new sample. For further details, as well as for an extensive analysis of the topic, we refer to the chapters about k-NN in [HTF01, SSBD14].

CHAPTER 3

Estimating g-Leakage via Machine Learning

In this chapter, as introduced in section 1.2, we propose a ML based method to perform black-box estimation of the g-vulnerability of a given system. We refer to section 2.1 for the mathematical background behind the notions of g-vulnerability and g-leakage which can be directly derived from the former provided some a priori knowledge about the distribution of the secrets.

We take inspiration from [CCP19], where ML based leakage estimation has been proposed for the first time. The authors show that the estimates based on ML algorithms tend to converge to the real value faster than methods based on estimating the internal conditional probabilities counting the frequencies of the outputs given some inputs (frequentist approach). This happens especially when the space of the observables is large, given the fact that, while the frequentist methods can only draw conclusions based on counts on the available samples, ML methods are, in general, able to extrapolate from the samples and provide better prediction (generalization) for the rest of the universe. The authors rely on the universally consistent k-NN rule[1] and the fact that the error of a classifier selected according to such a rule can be used for estimating the Bayes risk.

We improve on this by proposing to estimate the g-vulnerability which, given its flexibility, encompasses the concept of Bayes adversary and allows us to model different kinds of attacks. As anticipated in the introduction to this thesis, the idea is to reduce the problem to that of approximating the Bayes classifier, so that *any universally consistent ML algorithm can be used for the purpose*. This reduction essentially takes into account the impact of the gain function in the generation of the training data, and we propose two methods to obtain this effect, which we call *channel pre-processing* and *data pre-processing*, respectively. We evaluate our approach via experiments on various channels and gain functions. In order to show the generality of our approach, we use two different ML algorithms, namely k-NN and ANN, and we compare their performances. The experimental results show that

[1]The k-NN has the smallest possible probability of error when the number of samples $n_s \to \infty$, $k \to \infty$, and $k/n_s \to 0$. This has been proven in [Sto77].

Symbol	Description
$x \in \mathcal{X}$	a secret
$w \in \mathcal{W}$	a guess
$y \in \mathcal{Y}$	an observable output by the system
X	random variable for secrets, it takes values $x \in \mathcal{X}$
W	random variable for guesses, it takes values $w \in \mathcal{W}$
Y	random variable for observables, it takes values $y \in \mathcal{Y}$
$\lvert\mathcal{S}\rvert$	size of a set $\mathcal{S}$
$\mathcal{P}(\mathcal{S})$	Distribution over a set of symbols $\mathcal{S}$
$\mathcal{H}$	class of learning functions f
π, P_X	prior distribution over the secret space
$\hat{\pi}$, $\widehat{P}_X$	empirical prior distribution over the secret space
C	Channel matrix
$\pi \triangleright C$	joint distribution from prior π and channel C
P_{XY}	joint probability distribution
$\widehat{P}_{XY}$	empirical joint probability distribution
$P_{Y\lvert X}$	conditional probability of Y given X
$\widehat{P}_{Y\lvert X}$	empirical conditional probabilities
$\mathbb{P}$	probability measure
$\mathbb{E}[\cdot]$	expected value
$g(w, x)$	gain function that takes a guess w and secret x as inputs
G	gain matrix of size $\lvert\mathcal{W}\rvert \times \lvert\mathcal{X}\rvert$ according to a specific g
V_g	g-vulnerability
$V(f)$	g-vulnerability functional
$\widehat{V}_n(f)$	empirical g-vulnerability functional evaluated on n samples

Table 3.1 – Table of symbols for chapter 3.

our approach provides accurate estimations, and that, as expected, it outperforms by far the frequentist approach when the observables domain is large.

3.1 Learning g-vulnerability: Statistical Bounds

This section introduces the mathematical problem of learning g-vulnerability. More specifically, we address the problem of characterizing universal learnability in the present framework, and to this end, we derive distribution-free bounds on the accuracy of the estimation, implying statistical consistence of our estimator. Table 3.1 contains a summary of the most frequently used symbols in the following sections of this chapter.

3.1.1 Main definitions

We consider the problem of estimating the g-vulnerability from samples via ML models, and we show that the analysis of this estimation can be conducted in the general statistical framework of maximizing an expected functional using observed samples. The idea can be described using three components:

- A generator of random secrets $x \in \mathcal{X}$ with $|\mathcal{X}| < \infty$, drawn independently from a fixed but unknown distribution $P_X(x)$;
- a channel that returns an observable $y \in \mathcal{Y}$ with $|\mathcal{Y}| < \infty$ for every input x, according to a conditional distribution $P_{Y|X}(y|x)$, also fixed and unknown;
- a learning machine capable of implementing a set of rules $f \in \mathcal{H}$, where $\mathcal{H}$ denotes the class of functions $f : \mathcal{Y} \to \mathcal{W}$ and $\mathcal{W}$ is the finite set of guesses.

Moreover let us note that

$$g : \mathcal{W} \times \mathcal{X} \to [a, b] \tag{3.1}$$

for some finite real values $a \geqslant 0$ and $b > a$, and $\mathcal{X}$ and $\mathcal{W}$ are finite sets. The problem of learning the g-vulnerability is that of choosing the function $f : \mathcal{Y} \to \mathcal{W}$ which maximizes the functional $V(f)$, representing the expected gain, defined as:

$$V(f) \stackrel{\text{def}}{=} \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} g\big(f(y), x\big) P_{XY}(x, y). \tag{3.2}$$

Note that $f(y)$ corresponds to the "guess" w, for the given y, in eq. (2.2). The maximum of $V(f)$ is the g-vulnerability, namely:

$$V_g \stackrel{\text{def}}{=} \max_{f\in\mathcal{H}} V(f). \tag{3.3}$$

3.1.2 Principle of the empirical g-vulnerability maximization

Since we are in the black-box scenario, the joint probability distribution $P_{XY} \equiv \pi \triangleright C$ is unknown. We assume, however, the availability of m independent and identically distributed (i.i.d.) samples drawn according to P_{XY} that can be used as a training set $\mathcal{D}_m$ to solve the maximization of f over $\mathcal{H}$ and additionally n i.i.d. samples are available to be used as a validation[2] set $\mathcal{T}_n$ to estimate the average in eq. (3.2). Let us denote these sets as: $\mathcal{D}_m \stackrel{\text{def}}{=} \big\{(x_1, y_1), \ldots, (x_m, y_m)\big\}$ and $\mathcal{T}_n \stackrel{\text{def}}{=} \big\{(x_{m+1}, y_{m+1}), \ldots, (x_{m+n}, y_{m+n})\big\}$, respectively.

In order to maximize the g-vulnerability functional eq. (3.2) for an unknown probability measure P_{XY}, the following principle is usually applied. The expected g-vulnerability

[2]We prefer to call $\mathcal{T}_n$ validation set rather than test set, since we use it to estimate the g-vulnerability with the learned $f_m^\star$, rather than to measure the error in estimating the g-vulnerability.

functional $V(f)$ is replaced by the *empirical g-vulnerability functional*:

$$\widehat{V}_n(f) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{(x,y)\in\mathcal{T}_n} g\big(f(y), x\big), \tag{3.4}$$

which is evaluated on $\mathcal{T}_n$ rather than P_{XY}. This estimator is clearly unbiased in the sense that

$$\mathbb{E}\big[\widehat{V}_n(f)\big] = V(f).$$

Let $f_m^\star$ denote the empirical optimal rule given by

$$f_m^\star \stackrel{\text{def}}{=} \operatorname*{argmax}_{f\in\mathcal{H}} \widehat{V}_m(f),\ \ \widehat{V}_m(f) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{(x,y)\in\mathcal{D}_m} g\big(f(y), x\big), \tag{3.5}$$

which is evaluated on $\mathcal{D}_m$ rather than P_{XY}. The function $f_m^\star$ is the optimizer according to $\mathcal{D}_m$, namely the best way among the functions $f : \mathcal{Y} \to \mathcal{W}$ to approximate V_g by maximizing $\widehat{V}_m(f)$ over the class of functions $\mathcal{H}$. This principle is known in statistics as the Empirical Risk Maximization (ERM).

Intuitively, we would like $f_m^\star$ to give a good approximation of the g-vulnerability, in the sense that its expected gain

$$V(f_m^\star) \ = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} g\big(f_m^\star(y), x\big) P_{XY}(x,y) \tag{3.6}$$

should be close to V_g. Note that the difference

$$V_g - V(f_m^\star) = \max_{f\in\mathcal{H}} V(f) - V(f_m^\star) \tag{3.7}$$

is always non negative and represents the gap by selecting a possible suboptimal function $f_m^\star$. Unfortunately, we are not able to compute $V(f_m^\star)$ either, since P_{XY} is unknown and thus, eq. (3.7) cannot be measured in practice. In its place, we have to use its approximation $\widehat{V}_n(f_m^\star)$ and eq. (3.7) should be replaced by $V_g - \widehat{V}_n(f_m^\star)$ which is not always non-negative.

By using basics principles from statistical learning theory, we study two main questions:

- When does the estimator $\widehat{V}_n(f_m^\star)$ work? What are the conditions for its statistical consistency?

- How well does $\widehat{V}_n(f_m^\star)$ approximate V_g? In other words, how fast does the sequence of largest empirical g-leakage values converge to the largest g-leakage function? This is related to the so called rate of generalization of a learning machine that implements the ERM principle.

Auxiliary results

Before we move on to describing the bounds on the estimation accuracy, we would like to recall two main results from the literature, namely proposition 1 and lemma 1, that we will use for the proofs in the rest of the chapter.

Proposition 1 (Hoeffding's inequality [BLM13])**:** Let $Z_1, \ldots, Z_n$ be independent bounded random variables such that $Z_i \in [a_i, b_i]$ almost surely and let $S_n = \sum_{i=1}^n Z_i$. Then, for any $t > 0$, we have the following inequalities:

$$\mathbb{P}\big(S_n - \mathbb{E}[S_n] \geqslant t\big) \leqslant \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right), \tag{3.8}$$

$$\mathbb{P}\big(S_n - \mathbb{E}[S_n] \leqslant -t\big) \leqslant \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right). \tag{3.9}$$

Lemma 1: Let $r > 0$ and let Z be a real-valued random variable such that for all $t \geqslant 0$,

$$\mathbb{P}(Z \geqslant t) \leqslant 2q \exp\left(-\frac{t^2}{r^2}\right). \tag{3.10}$$

Then, for $q > 1$,

$$\mathbb{E}[Z] \leqslant r\left(\sqrt{\ln q} + \frac{1}{\sqrt{\ln q}}\right) \tag{3.11}$$

and for $q = 1$,

$$\mathbb{E}[Z] \leqslant \sqrt{2} r. \tag{3.12}$$

Proof.

$$\begin{aligned}
\mathbb{E}[Z] &= \int_0^\infty \mathbb{P}(Z \geqslant t) dt \\
&\leqslant \int_0^{r\sqrt{\ln q}} \mathbb{P}(Z \geqslant t) dt + \int_{r\sqrt{\ln q}}^\infty 2q \exp\left(-\frac{t^2}{r^2}\right) dt \\
&\leqslant r\sqrt{\ln q} + \int_{r\sqrt{\ln q}}^\infty 2q \exp\left(-\frac{t^2}{r^2}\right) dt \\
&\leqslant r\sqrt{\ln q} + 2q \int_{r\sqrt{\ln q}}^\infty \frac{t}{r\sqrt{\ln q}} \exp\left(-\frac{t^2}{r^2}\right) dt \\
&= r\sqrt{\ln q} + \frac{2q}{r\sqrt{\ln q}} \frac{r^2}{2} \exp\left(-\frac{(r\sqrt{\ln q})^2}{r^2}\right) \\
&= r\left(\sqrt{\ln q} + \frac{1}{\sqrt{\ln q}}\right).
\end{aligned}$$

Similarly, the second statement holds if $q = 1$ and the integral is split between $t \in [0, r]$ and $t \in [r, \infty)$. □

3.1.3 Distribution-free bounds on the estimation accuracy

We start with the following lemma which is a simple adaption of the uniform deviations of relative frequencies from probabilities theorems in [DGL96].

Lemma 2: The following inequalities hold:

$$V_g - V(f_m^\star) \leqslant 2 \max_{f \in \mathcal{H}} \left| \widehat{V}_m(f) - V(f) \right|, \tag{3.13}$$

$$\left| \widehat{V}_n(f_m^\star) - V(f_m^\star) \right| \leqslant \max_{f \in \mathcal{H}} \left| \widehat{V}_n(f) - V(f) \right|. \tag{3.14}$$

Proof. Recall the definition of $f_m^\star$ in eq. (3.5) and let $f^\star = \operatorname{argmax}_{f \in \mathcal{H}} V(f)$. We first observe that

$$\left| \widehat{V}_n(f') - V(f') \right| \leqslant \max_{f \in \mathcal{H}} \left| \widehat{V}_n(f) - V(f) \right|$$

for all $f' \in \mathcal{H}$. Inequality eq. (3.14) follows by letting $f' = f_m^\star$. We now prove expression eq. (3.13):

$$\begin{aligned}
V_g - V(f_m^\star) &= V(f^\star) - \widehat{V}_m(f_m^\star) + \widehat{V}_m(f_m^\star) - V(f_m^\star) \\
&\leqslant V(f^\star) - \widehat{V}_m(f_m^\star) + \left| \widehat{V}_m(f_m^\star) - V(f_m^\star) \right| && (3.15) \\
&\leqslant V(f^\star) - \widehat{V}_m(f^\star) + \left| \widehat{V}_m(f_m^\star) - V(f_m^\star) \right| && (3.16) \\
&\leqslant \left| V(f^\star) - \widehat{V}_m(f^\star) \right| + \left| \widehat{V}_m(f_m^\star) - V(f_m^\star) \right| && (3.17) \\
&\leqslant \max_{f \in \mathcal{H}} \left| \widehat{V}_m(f) - V(f) \right| + \max_{f \in \mathcal{H}} \left| \widehat{V}_m(f) - V(f) \right| && (3.18) \\
&\leqslant 2 \max_{f \in \mathcal{H}} \left| \widehat{V}_m(f) - V(f) \right|. && (3.19)
\end{aligned}$$

□

The above lemma implies that $\max_{f \in \mathcal{H}} \left| \widehat{V}_m(f) - V(f) \right|$ and $\max_{f \in \mathcal{H}} \left| \widehat{V}_n(f) - V(f) \right|$ provide upper bounds for two deviations:

- the suboptimality of $f_m^\star$ learned using the training set and the class $\mathcal{H}$, that is, how large $V_g - V(f_m^\star)$ is;
- the estimation error $\left| \widehat{V}_n(f_m^\star) - V(f_m^\star) \right|$ due to the use of a validation set instead of the true expected gain $V(f_m^\star)$, for the selected function $f_m^\star$.

Next let us remind the reader of the fact that $g : \mathcal{W} \times \mathcal{X} \to [a, b]$ for some finite real values $a \geqslant 0$ and $b > a$, and $\mathcal{X}$ and $\mathcal{W}$ are finite sets. This is important as it allows to probabilistically delimit the bounds of eq. (3.13) and eq. (3.14) in terms of $b - a$, besides the size n of the validation set $\mathcal{T}_n$, as expressed by the following proposition.

Proposition 2 (Uniform deviations)**:** Assume that $|\mathcal{H}| < \infty$ and $g : \mathcal{W} \times \mathcal{X} \longrightarrow [a, b]$, for a, b real values such that $a \geqslant 0$ and $b > a$. Then, we have for all $\varepsilon > 0$,

$$\sup_{P_{XY}} \mathbb{P}\left(\left|\widehat{V}_n(f_m^\star) - V(f_m^\star)\right| > \varepsilon\right) \leqslant 2\exp\left(-\frac{2n\varepsilon^2}{(b-a)^2}\right) \tag{3.20}$$

and

$$\sup_{P_{XY}} \mathbb{P}\left(V_g - V(f_m^\star) > \varepsilon\right) \leqslant 2|\mathcal{H}|\exp\left(-\frac{m\varepsilon^2}{2(b-a)^2}\right). \tag{3.21}$$

Proof. We first prove expression eq. (3.20). Notice that

$$\begin{aligned}
&\mathbb{P}\left(\left|\widehat{V}_n(f_m^\star) - V(f_m^\star)\right| > \varepsilon\right) \\
&= \mathbb{E}_{\mathcal{D}_m \sim P_{XY}^m} \mathbb{P}\left(\left|\widehat{V}_n(f_m^\star) - V(f_m^\star)\right| > \varepsilon \mid \mathcal{D}_m\right) &(3.22)\\
&\leqslant 2\exp\left(-\frac{2n\varepsilon^2}{(b-a)^2}\right), &(3.23)
\end{aligned}$$

where eq. (3.22) follows by conditioning on the training samples and then taking the probability on the validation samples, and eq. (3.23) follows by noticing that given the training samples the function $f_m^\star$ is fixed by applying proposition 1 the inequality follows. The second inequality in eq. (3.21) is a consequence of the following steps:

$$\begin{aligned}
\mathbb{P}\left(V_g - V(f_m^\star) > \varepsilon\right) &\leqslant \mathbb{P}\left(\max_{f\in\mathcal{H}} \left|\widehat{V}_m(f) - V(f)\right| > \varepsilon/2\right) &(3.24)\\
&= \mathbb{P}\left(\bigcup_{f\in\mathcal{H}} \left\{\left|\widehat{V}_m(f) - V(f)\right| > \varepsilon/2\right\}\right) &(3.25)\\
&\leqslant \sum_{f\in\mathcal{H}} \mathbb{P}\left(\left|\widehat{V}_m(f) - V(f)\right| > \varepsilon/2\right) &(3.26)\\
&\leqslant 2|\mathcal{H}|\exp\left(-\frac{2m\varepsilon^2}{4(b-a)^2}\right), &(3.27)
\end{aligned}$$

where eq. (3.24) follows by applying the first inequality in lemma 2 and eq. (3.27) follows from proposition 1 with an appropriate redefinition to $\varepsilon/2$. □

Expression eq. (3.20) shows that the estimation error due to the use of a validation set in $\widehat{V}_n(f_m^\star)$ instead of the true expected gain $V(f_m^\star)$ vanishes with the number of validation samples. On the other hand, expression eq. (3.21) implies 'learnability' of an optimal f, i.e., the suboptimality of $f_m^\star$ learned using the training set $\widehat{V}_m(f_m^\star)$ vanishes with the number of training samples. Both expressions toguether imply that the g-vulnerability is learnable for all distributions (data sets) via the ERM principle introduced in eq. (3.4). In other words, whenever the bounds indicate that we are close to the optimum f, we must at the same time

have a good estimate of the g-vulnerability, and vice versa. Although the bound in eq. (3.21) is in perfect agreement with the long-standing results from statistical learning theory which show that learnability is equivalent to uniform convergence of the empirical risk eq. (3.5) to the actual risk eq. (3.2), this distribution-free bound is rather pessimistic and cannot be expected to predict the performance in practical scenarios.

We can now state the main result of this section, namely an upper bound on the average estimation error of g-vulnerability.

Theorem 1: The averaged estimation error of the g-vulnerability can be bounded as follows:

$$\mathbb{E}\big|V_g - \widehat{V}_n(f_m^\star)\big| \leqslant V_g - \mathbb{E}\big[V(f_m^\star)\big] + \mathbb{E}\big|V(f_m^\star) - \widehat{V}_n(f_m^\star)\big|,$$

where the expectations are understood over all possible training and validation sets drawn according to P_{XY}. Furthermore,

$$V_g - \mathbb{E}\big[V(f_m^\star)\big] \leqslant \sqrt{\frac{2(b-a)^2}{m}}\left(\sqrt{\ln|\mathcal{H}|} + \frac{1}{\sqrt{\ln|\mathcal{H}|}}\right), \tag{3.28}$$

$$\mathbb{E}\big|V(f_m^\star) - \widehat{V}_n(f_m^\star)\big| \leqslant \sqrt{\frac{(b-a)^2}{n}}, \tag{3.29}$$

independently of the specific underlying distribution P_{XY}.

Proof. Observe that

$$\begin{aligned}\mathbb{E}\big|V_g - \widehat{V}_n(f_m^\star)\big| &= \mathbb{E}\big|V_g - V(f_m^\star) + V(f_m^\star) - \widehat{V}_n(f_m^\star)\big| \\ &\leqslant V_g - \mathbb{E}\big[V(f_m^\star)\big] + \mathbb{E}\big|V(f_m^\star) - \widehat{V}_n(f_m^\star)\big|,\end{aligned}$$

which follows from the triangular inequality. We first bound the second term in the previous inequality as follows:

$$\mathbb{E}\big|V(f_m^\star) - \widehat{V}_n(f_m^\star)\big| \leqslant \sqrt{\frac{(b-a)^2}{n}}, \tag{3.30}$$

where the claim in eq. (3.30) follows by applying proposition 2 and using expression eq. (3.20) combined with Lemma lemma 1 choosing $q = 1$ and $r^2 = \frac{(b-a)^2}{2n}$.

We now bound the first term. Notice that

$$\mathbb{P}\left(V_g - V(f_m^\star) > \varepsilon\right) \leqslant 2|\mathcal{H}| \exp\left(-\frac{m\varepsilon^2}{2(b-a)^2}\right), \tag{3.31}$$

where eq. (3.31) follows from inequality eq. (3.21) in proposition 2. By using again lemma 1 with $q = |\mathcal{H}|$ and $r^2 = \frac{2(b-a)^2}{m}$, we obtain

$$V_g - \mathbb{E}\big[V(f_m^\star)\big] \leqslant \sqrt{\frac{2(b-a)^2}{m}}\left(\sqrt{\log|\mathcal{H}|} + \frac{1}{\sqrt{\log|\mathcal{H}|}}\right), \tag{3.32}$$

which concludes the proof. □

Interestingly, the term corresponding to eq. (3.28) is the error induced when estimating the function $f_m^\star$ using m samples from the training set while eq. (3.29) indicates the error incurred when estimating the g-vulnerability using n samples from the validation set. Clearly, the scaling of these bounds with the number of samples are very different which can be made evident by using the order notation:

$$\sup_{P_{XY}} \left\{ V_g - \mathbb{E}\big[V(f_m^\star)\big] \right\} \equiv \mathcal{O}\left(\sqrt{\frac{|\mathcal{Y}| \ln |\mathcal{W}|}{m}} \right), \tag{3.33}$$

$$\sup_{P_{XY}} \mathbb{E}\big| V(f_m^\star) - \widehat{V}_n(f_m^\star) \big| \equiv \mathcal{O}\left(\frac{1}{\sqrt{n}} \right). \tag{3.34}$$

These distribution-free bounds indicate that the error in eq. (3.34) vanishes much faster than the error in eq. (3.33) and thus, the size of the training set, in general, should be kept larger than the size of the validation set, i.e., $n < m$. However, the bound in eq. (3.33) is rather pessimistic since it suffers from being independent of the underlying distribution and the optimization method used to solve eq. (3.4). Tighter bounds can be derived but they would require statistical knowledge of the data-generating distribution which is often not available in real-world scenarios.

3.1.4 Sample complexity

We now study how large the validation set should be in order to get a good estimation. For $\varepsilon, \delta > 0$, we define the sample complexity as the set of smallest integers $M(\varepsilon, \delta)$ and $N(\varepsilon, \delta)$ sufficient to guarantee that the gap between the true g-vulnerability and the estimated $\widehat{V}_n(f_m^\star)$ is at most ε with at least $1 - \delta$ probability:

Definition 1: For $\varepsilon, \delta > 0$, let all pairs $\big(M(\varepsilon, \delta), N(\varepsilon, \delta)\big)$ be the set of smallest (m, n) sizes of training and validation sets such that:

$$\sup_{P_{XY}} \mathbb{P}\left[|V_g - \widehat{V}_n(f_m^\star)| > \varepsilon \right] \leqslant \delta. \tag{3.35}$$

Next result says that we can bound the sample complexity in terms of ε, δ, and $|b - a|$.

Corollary 1: The sample complexity of the ERM algorithm g-vulnerability is bounded from above by the set of values satisfying:

$$M(\varepsilon, \delta) \leqslant \frac{2(b-a)^2}{\varepsilon^2} \ln\left(\frac{2|\mathcal{H}|}{\delta - \Delta} \right), \tag{3.36}$$

$$N(\varepsilon, \delta) \leqslant \frac{(b-a)^2}{2\varepsilon^2} \ln\left(\frac{2}{\Delta} \right), \tag{3.37}$$

for all $0 < \Delta < \delta$.

Proof. We first notice that

$$\mathbb{P}\left(|V_g - \widehat{V}_n(f_m^\star)| > \varepsilon\right) \leqslant \mathbb{P}\left(V_g - V(f_m^\star) > \varepsilon\right) + \mathbb{P}\left(|V(f_m^\star) - \widehat{V}_n(f_m^\star)| > \varepsilon\right), \tag{3.38}$$

and thus from eq. (3.20) and eq. (3.21) in proposition 2, we have

$$\mathbb{P}\left(|V_g - \widehat{V}_n(f_m^\star)| > \varepsilon\right) \leqslant 2\exp\left(-\frac{2n\varepsilon^2}{(b-a)^2}\right) + 2|\mathcal{H}|\exp\left(-\frac{m\varepsilon^2}{2(b-a)^2}\right). \tag{3.39}$$

Let us require:

$$2|\mathcal{H}|\exp\left(-\frac{m\varepsilon^2}{2(b-a)^2}\right) \leqslant (\delta - \Delta), \tag{3.40}$$

$$2\exp\left(-\frac{2n\varepsilon^2}{(b-a)^2}\right) \leqslant \Delta, \tag{3.41}$$

which satisfies the desired condition:

$$\mathbb{P}\left(|V_g - \widehat{V}_n(f_m^\star)| > \varepsilon\right) \leqslant \delta, \tag{3.42}$$

for any $0 < \Delta < \delta$. Finally, from the previous inequality we can derive lower bounds on n and m:

$$m \geqslant \frac{2(b-a)^2}{\varepsilon^2}\ln\left(\frac{2|\mathcal{H}|}{\delta - \Delta}\right), \tag{3.43}$$

$$n \geqslant \frac{(b-a)^2}{2\varepsilon^2}\ln\left(\frac{2}{\Delta}\right), \tag{3.44}$$

which by definition of sample complexity shows the corollary. □

The theoretical results of this section are very general and do not refer to any particular model or data distribution. In particular, it is important to emphasize that the upper bounds in eq. (3.13) and eq. (3.14) are independent of the learned function $f_m^\star$, and thus they are independent of the specific algorithm and training sets in used to solve the optimization in eq. (3.5). Furthermore, the f maximizing $|V(f) - \widehat{V}_n(f)|$ in those in-equations is not necessarily what the algorithm would choose. Hence the bounds given in theorem 1 and corollary 1 in general are not tight. However, these theoretical bounds provide a worst-case measure from which learnability holds for all data sets.

In the next section, we will propose an approach for selecting $f_m^\star$ and estimating V_g. The experiments in section 3.3 suggest that our method usually estimates V_g much more accurately than what is indicated by theorem 1.

3.2 From g-vulnerability to Bayes vulnerability via pre-processing

This is the core section of this chapter, where we describe our approach to select the $f_m^\star$ to estimate V_g.

In principle one could train a neural network to learn $f_m^\star$ by using $-\widehat{V}_m(f)$ as the loss function, and minimizing it over the m training samples (cfr. eq. (3.5)). However, this would require $\widehat{V}_m(f)$ to be a differentiable function of the weights of the neural network, so that its gradient can be computed during the back-propagation. Now, the problem is that the g component of $\widehat{V}_m(f)$ is essentially a non-differentiable function, so it would need to be approximated by a suitable differentiable (surrogate) function, (e.g., as it is the case of the Bayes error via the cross-entropy). Finding an adequate differentiable function to replace each possible g may be a challenging task in practice. If this surrogate does not preserve the original dynamic of the gradient of g with respect to f, the learned f will be far from being optimal.

In order to circumvent this issue, we propose a different approach, which presents two main advantages:

1. it reduces the problem of learning $f_m^\star$ to a standard classification problem, therefore it does not require a different loss function to be implemented for each adversarial scenario;

2. it can be implemented by using *any* universally consistent learning algorithm (i.e., any ML algorithm approximating the ideal Bayes classifier).

The reduction described in the above list (item 1) is based on the idea that, in the g-leakage framework, the adversary's goal is not to directly infer the actual secret x, but rather to select the optimal guess w about the secret. As a consequence, the training of the ML classifier to produce $f_m^\star$ should not be done on pairs of type (x, y), but rather of type (w, y), expressing the fact that the best guess, in the particular run which produced y, is w. This shift from (x, y) to (w, y) is via a *pre-processing* and we propose two distinct and systematic ways to perform this transformation, called *data* and *channel pre-processing*, respectively. The two methods are illustrated in the following sections.

We remind that, according to section 3.1, we restrict, without loss of generality, to non-negative g's. If g takes negative values, then it can be shifted by adding $-\min_{w,x} g(w, x)$, without consequences for the g-leakage value (cfr. [ACPS12, ACM+14]). Furthermore we assume that there exists at least a pair (x, w) such that $\pi_x \cdot g(w, x) > 0$. Otherwise V_g would be 0 and the problem of estimating it will be trivial.

3.2.1 Data pre-processing

The data pre-processing technique is completely black-box in the sense that it does not need access to the channel. We only assume the availability of a set of pairs of type (x, y), sampled according to $\pi \triangleright C$, the input-output distribution of the channel. This set could be provided by

Algorithm 1: Algorithm for data pre-processing

Input: $\mathcal{D}_m$; *Output*: $\mathcal{D}'_{m'}$;
1. $\mathcal{D}'_{m'} := \varnothing$;
2. For each x, y, let u_{xy} be the number of copies of (x, y) in $\mathcal{D}_m$;
3. For each x, y, w, add $u_{xy} \cdot g(w, x)$ copies of (w, y) to $\mathcal{D}'_{m'}$.

a third party, for example. We divide the set in $\mathcal{D}_m$ (training) and $\mathcal{T}_n$ (validation), containing m and n pairs, respectively.

For the sake of simplicity, to describe this technique we assume that g takes only integer values, in addition to being non-negative.

The idea behind the data pre-processing technique is that the effect of the gain function can be represented in the transformed dataset by amplifying the impact of the guesses in proportion to their reward. For example, consider a pair (x, y) in $\mathcal{D}_m$, and assume that the reward for the guess w is $g(w, x) = 5$, while for another guess w' is $g(w', x) = 1$. Then in the transformed dataset $\mathcal{D}'_{m'}$ this pair will contribute with 5 copies of (w, y) and only 1 copy of (w', y). The transformation is described in algorithm 1. Note that in general it causes an expansion of the original dataset.

Estimation of V_g

Given $\mathcal{D}_m$, we construct the set $\mathcal{D}'_{m'}$ of pairs (w, y) according to Algorithm algorithm 1. Then, we use $\mathcal{D}'_{m'}$ to train a classifier $f^\star_{m'}$, using an algorithm that approximates the ideal Bayes classifier. As proved below, $f^\star_{m'}$ gives the same mapping $\mathcal{Y} \to \mathcal{W}$ as the optimal empirical rule $f^\star_m$ on $\mathcal{D}_m$ (cfr. section 3.1.2). Finally, we use $f^\star_m$ and $\mathcal{T}_n$ to compute the estimation of $V_g(\pi, C)$ as in eq. (3.4), with f replaced by $f^\star_m$.

Correctness

Let us first introduce some notation. For each (w, y), define:

$$U(w, y) \stackrel{\text{def}}{=} \sum_x \pi_x \cdot C_{xy} \cdot g(w, x) \,, \tag{3.45}$$

which represents the "ideal" proportion of copies of (w, y) that $\mathcal{D}'_{m'}$ should contain (of course, such proportion is only approximated in $\mathcal{D}'_{m'}$ since it is generated from $\mathcal{D}_m$ rather than according to the true distribution $\pi \triangleright C$). From $U(w, y)$ we can now derive the ideal joint distribution on $\mathcal{W} \times \mathcal{Y}$ and the marginal on $\mathcal{W}$:

$$P_{WY}(w, y) \stackrel{\text{def}}{=} \frac{U(w, y)}{\alpha}, \quad \text{where} \quad \alpha \stackrel{\text{def}}{=} \sum_{y,w} U(w, y) \,, \tag{3.46}$$

(note that $\alpha > 0$ because of the assumption on π and g),

$$\sigma_w \stackrel{\text{def}}{=} \sum_y P_{WY}(w, y). \tag{3.47}$$

The channel of the conditional probabilities of y given w is:

$$E_{wy} \stackrel{\text{def}}{=} \frac{P_{WY}(w,y)}{\sigma_w} \,. \tag{3.48}$$

Note that $P_{WY} = \sigma \triangleright E$. By construction, it is clear that the $\mathcal{D}'_{m'}$ generated by Algorithm algorithm 1 could have been generated, with the same probability, by sampling $\sigma \triangleright E$. The following theorem establishes that the g-vulnerability of $\pi \triangleright C$ is equivalent to the Bayes vulnerability of $\sigma \triangleright E$, and therefore that it is correct to estimate $f^\star_m$ as an empirical Bayes classifier $f^\star_{m'}$ trained using $\mathcal{D}'_{m'}$.

Theorem 2 (Correctness of data pre-processing)**:** Given a prior π, a channel C, and a gain function g, we have

$$V_g(\pi, C) \; = \alpha \cdot V_{g_{\text{id}}}(\sigma, E) \,,$$

where α, σ and E are those defined in eq. (3.46), eq. (3.47) and eq. (3.48), respectively, and g_{id} is the identity function (cfr. section 2.1), i.e., the gain function corresponding to the Bayesian adversary.

Proof.

$$\begin{aligned}
V_{g_{\text{id}}}(\sigma, E) &= \sum_y \max_w \sum_{w'} \sigma_{w'} \cdot E_{w'y} \cdot g_{\text{id}}(w, w') \\
&= \sum_y \max_w (\sigma_w E_{wy}) \\
&= \sum_y \max_w P_{WY}(w, y) \\
&= \sum_y \max_w \tfrac{U(w,y)}{\alpha} \\
&= \tfrac{1}{\alpha} \cdot \sum_y \max_w \textstyle\sum_x \pi_x \cdot C_{xy} \cdot g(w, x) \\
&= \tfrac{1}{\alpha} \cdot V_g(\pi, C)
\end{aligned}$$

□

The α in the above theorem is only a scale factor, hence it does not influence the selection of $f^\star_m$. Note that an alternative way to estimate $V_g(\pi, C)$ would be by computing the empirical Bayes error of $f^\star_{m'}$ (using $V_{g_{\text{id}}}(\sigma, E)$) on a validation set $\mathcal{T}'_{n'}$ of type (w, y) generated from $\mathcal{T}_n$ by the same transformation as from $\mathcal{D}_m$ to $\mathcal{D}'_{m'}$. In this case the α would be necessary for converting the estimation of $V_{g_{\text{id}}}(\sigma, E)$ into the estimation of $V_g(\pi, C)$.

General case: data pre-processing when g does not take integer values

Approximating g so that it only takes values $\in \mathbb{Q}_{\geqslant 0}$ allows us to represent each gain as a quotient of two integers, namely $\text{Numerator}(G_{w,x})/\text{Denominator}\,(G_{w,x})$. Let us also define

$$K \stackrel{\text{def}}{=} \text{lcm}_{wx}(\text{Denominator}(G_{w,x})), \tag{3.49}$$

where $\text{lcm}(\cdot)$ is the least common multiple. Multiplying G by K gives the integer version of the gain matrix that can replace the original one. It is clear that the calculation of the least common multiplier, as well as the increase in the amount of data produced during the dataset building using a gain matrix forced to be integer, might constitute a relevant computational burden.

3.2.2 Channel pre-processing

For this technique we assume *black-box access* to the system, meaning that, although its channel matrix C is unknown, we can execute the system while controlling each input, and collect the corresponding output.

The core idea behind this technique is to transform the input of C into entries of type w, and to ensure that the distribution on the w's reflects the corresponding rewards expressed by g.

More formally, let us define a distribution τ on $\mathcal{W}$ as follows:

$$\tau_w \stackrel{\text{def}}{=} \frac{\sum_x \pi_x \cdot g(w,x)}{\beta} \quad \text{where} \quad \beta \stackrel{\text{def}}{=} \sum_{x,w} \pi_x \cdot g(w,x)\,, \tag{3.50}$$

(note that β is strictly positive because of the assumptions on g and π), and let us define the following matrix R from $\mathcal{W}$ to $\mathcal{X}$:

$$R_{wx} \stackrel{\text{def}}{=} \frac{1}{\beta} \cdot \frac{1}{\tau_w} \cdot \pi_x \cdot g(w,x)\,. \tag{3.51}$$

It is easy to check that R is a stochastic matrix, hence the composition RC is a channel. It is important to emphasize the following:

REMARK In the above definitions, β, τ and R depend solely on g and π, and not on C.

The above property is crucial to our goals, because in the black-box approach we are not supposed to rely on the knowledge of C's internals. We now illustrate how we can estimate V_g using the pre-processed channel RC.

Estimation of V_g

Given RC and τ, we build a set $\mathcal{D}''_{m''}$ consisting of pairs of type (w,y) sampled from $\tau \triangleright RC$. We also construct a set $\mathcal{T}_n$ of pairs of type (x,y) sampled from $\pi \triangleright C$. Then, we use $\mathcal{D}''_{m''}$ to

train a classifier $f^\star_m$, using an algorithm that approximates the ideal Bayes classifier. Finally, we use $f^\star_m$ and $\mathcal{T}_n$ to compute the estimation of $V_g(\pi, C)$ as in eq. (3.4), with f replaced by $f^\star_m$.

Alternatively, we could estimate $V_g(\pi, C)$ by computing the empirical Bayes error of $f^\star_m$ on a validation set $\mathcal{T}_n$ of type (w, y) sampled from $\tau \triangleright RC$, but the estimation would be less precise. Intuitively, this is because RC is more "noisy" than C.

Correctness

The correctness of the channel pre-processing method is given by the following theorem, which shows that we can learn $f^\star_m$ by training a Bayesian classifier on a set sampled from $\tau \triangleright RC$.

Theorem 3 (Correctness of channel pre-processing)**:** Given a prior π and a gain function g, we have that, for any channel C:

$$V_g(\pi, C) \;= \beta \cdot V_{g_{\mathrm{id}}}(\tau, RC) \quad \text{for all channels } C.$$

where β, τ and R are those defined in eq. (3.50) and eq. (3.51).

Proof. In this proof we use a notation that highlights the structure of the preprocessing. We will denote by G be the matrix form of g, i.e., $G_{wx} = g(w, x)$, and by Ψ^π the square matrix with π in its diagonal and 0 elsewhere. We have that $\beta = \|G\Psi^\pi\|_1 = \sum_{w,x} G_{wx}\pi_x$, which is strictly positive because of the assumptions on g and π. Furthermore, we have

$$\tau^T = \beta^{-1} G\Psi^\pi \mathbf{1}\,, \qquad R = \beta^{-1}(\Psi^\tau)^{-1} G\Psi^\pi\,,$$

where $\mathbf{1}$ is the vector of 1s and τ^T represents the transposition of vector τ. Note that $(\Psi^\tau)^{-1}$ is a diagonal matrix with entries τ_w^{-1} in its diagonal. If $\tau_w = 0$ then the row $R_{w,\cdot}$ is not properly defined; but its choice does not affect $V_{g_{\mathrm{id}}}(\tau, RC)$ since the corresponding prior is 0; so we can choose $R_{w,\cdot}$ arbitrarily (or equivalently remove the action w, it can never be optimal since it gives 0 gain). It is easy to check that τ is a proper distribution and R is a proper channel:

$$\begin{aligned} \textstyle\sum_w \tau_w = \sum_w \beta^{-1} \sum_x G_{wx}\pi_x = \beta^{-1}\beta = 1\,, \\ \textstyle\sum_x R_{w,x} = \sum_x \frac{1}{\tau_w}\beta^{-1} G_{wx}\pi_x = \frac{\tau_w}{\tau_w} = 1\,. \end{aligned}$$

Moreover, it holds that:

$$\beta\Psi^\tau R \;=\; \beta\Psi^\tau \beta^{-1}\Psi^{\tau\,-1} G\Psi^\pi \;=\; G\Psi^\pi\,.$$

The main result follows from the trace-based formulation of the posterior g-vulnerability [ACPS12], since for any channel C and strategy S, the above equation directly yields

$$\begin{aligned} V_g(\pi, C) \;&=\; \max_S \mathbf{tr}(G\Psi^\pi CS) \\ &=\; \beta \cdot \max_S \mathbf{tr}(\Psi^\tau RCS) \\ &=\; \beta \cdot V_{g_{\mathrm{id}}}(\tau, RC)\,, \end{aligned}$$

where $\mathbf{tr}(\cdot)$ is the matrix trace. □

Interestingly, a result similar to theorem 3 is also given in [BS16], although the context is completely different from ours: the focus of [BS16], indeed, is to study how the leakage of C on X may induce also a leakage of other sensitive information Z that has nothing to do with C (in the sense that is not information manipulated by C). We intend to explore this connection in the context of a possible extension of our approach to this more general scenario.

3.2.3 Data and channel pre-processing: pros and cons

The fundamental advantage of data pre-processing is that it allows to estimate V_g from just samples of the system, without even black-box access. In contrast to channel pre-processing, however, this method is particularly sensitive to the values of the gain function g. Large gain values will increase the size of $\mathcal{D}'_{m'}$, with consequent increase of the computational cost for estimating the g-vulnerability. Moreover, if g takes real values then we need to apply the technique described in section 3.2.1, which can lead to a large increase of the dataset as well. In contrast, the channel pre-processing method has the advantage of controlling the size of the training set, but it can be applied only when it is possible to interact with the channel by providing input and collecting output. Finally, from the precision point of view, we expect the estimation based on data pre-processing to be more accurate when g consists of small integers, because the channel pre-processing introduces some extra noise in the channel.

3.3 Evaluation

In this section we evaluate our approach to the estimation of g-vulnerability. We consider four different scenarios:

1. $\mathcal{X}$ is a set of (synthetic) numeric data, the channel C consists of *geometric noise*, and g is the *multiple guesses* gain function, representing an adversary that is allowed to make several attempts to discover the secret.

2. $\mathcal{X}$ is a set of locations from the Gowalla dataset [LK], C is the optimal noise of Shokri et al. [STT+12b], and g is one of the functions used to evaluate the privacy loss in [STT+12b], namely a function anti-monotonic on the distance, representing the idea that the more the adversary's guess is close to the target (i.e., the real location), the more he gains.

3. $\mathcal{X}$ is the Cleveland heart disease dataset [DG17], C is a *differentially private mechanism* [DMNS06, Dwo06], and g is a function that assigns higher values to worse heart conditions, modeling an adversary that aims at discovering whether a patient is at risk (for instance, to deny his application for health insurance).

4. $\mathcal{X}$ is a set of passwords of 128 bits and C is a *password checker* that leaks the time before the check fails, but mitigates the timing attacks by applying some random delay

and the bucketing technique (see, for example, [KD09]). The function g represents the part of the password under attack.

For each scenario, we proceed in the following way:

- We consider 3 different samples sizes for the training sets that are used to train the ML models and learn the $\mathcal{Y} \to \mathcal{W}$ remapping. This is to evaluate how the precision of the estimate depends on the amount of data available, and on its relation with the size of $|\mathcal{Y}|$.
- In order to evaluate the variance of the precision, for each size we create 5 different training sets, and
- for each trained model we estimate the g-vulnerability using 50 different validation sets.

3.3.1 Representation of the results and metrics

We graphically represent the results of the experiment as box plots, using one box for each size. More precisely, given a specific size, let $\widehat{V}_n^{ij}$ be the g-vulnerability estimation on the j-th validation set computed with a model trained over the i-th training set (where $i \in \{1, \ldots, 5\}$ and $j \in \{1, \ldots, 50\}$). Let V_g be the real g-vulnerability of the system. We define the *normalized estimation error* δ_{ij} and the mean value $\overline{\delta}$ of the δ_{ij}'s as follows:

$$\delta_{ij} \stackrel{\text{def}}{=} \frac{|\widehat{V}_n^{ij} - V_g|}{V_g}, \quad \text{with} \quad \overline{\delta} \stackrel{\text{def}}{=} \frac{1}{250} \sum_{i=1}^{5} \sum_{j=1}^{50} \delta_{ij} \,. \tag{3.52}$$

In the graphs, the δ_{ij}'s are the values reported in the box corresponding to the given size, and $\overline{\delta}$ is the black horizontal line inside the box.

We also consider the following quantities, which are typical measures of precision:

$$\text{dispersion} \stackrel{\text{def}}{=} \sqrt{\frac{1}{250} \sum_{i=1}^{5} \sum_{j=1}^{50} (\delta_{ij} - \overline{\delta})^2} \,, \tag{3.53}$$

$$\text{total error} \stackrel{\text{def}}{=} \sqrt{\frac{1}{250} \sum_{i=1}^{5} \sum_{j=1}^{50} \delta_{ij}^2} \,. \tag{3.54}$$

The dispersion is an average measure of how far the normalized estimation errors are from their mean value when using same-size training and validation sets. On the other hand, the total error is an average measure of the normalized estimation error, when using same-size training and validation sets. In our experiments we will see that, as expected, the dispersion and the total error tend to decrease when the amount of training samples increases.

In order to make a fair comparison between the two presented pre-processing methods, intuitively we need to train them on training sets of the same size, and evaluate them on validation sets of the same size. For the validation part, since the best f function has been already found and therefore we do not need any data pre-processing, this is easy to guarantee. But what does "same size" mean, in the case of training sets built with different pre-processing methods? Consider the following situation: assume that we have a set of data $\mathcal{D}_m$ coming from a third party collector (we recall that the index m represents the size of the set), and let $\mathcal{D}'_{m'}$ be the result of the data pre-processing on $\mathcal{D}_m$. Now, let $\mathcal{D}''_{m''}$ be the dataset obtained drawing samples according to the channel pre-processing method. Should we impose $m'' = m$ or $m'' = m'$? We argue that the right choice is the first one, because the main limiting constraint in our method is the amount of "real" data that we can collect, one way or another. In case we cannot interact with the channel, the number of available samples is controlled by the data provider. Indeed, $\mathcal{D}'_{m'}$ is generated synthetically from $\mathcal{D}_m$ and cannot contain more information about C than $\mathcal{D}_m$, despite its larger size.

The nice feature of the normalized estimation error is that, thanks to the normalization, it allows to compare the results among different scenario and different levels of (real) g-vulnerability. Also, the error is more meaningful than the absolute value.

3.3.2 Learning algorithms

We consider two ML algorithms in the experiments: the k-Nearest Neighbors (k-NN) and the Artificial Neural Networks (ANN). We have made however a slight modification of k-NN algorithm, due to the following reason: recall that, depending on the particular gain function, the data pre-processing method might create many instances where a certain observable y is repeated multiple times in pair with different w's. For the k-NN algorithm, a very common choice is to consider a number of neighbors which is equivalent to natural logarithm of the total number of training samples. In particular, when the data pre-processing is applied, this means that $k = \log(m')$ nearest neighbors will be considered for the classification decision. Since $\log(m')$ grows slowly with respect to m', it might happen that k-NN fails to find the subset of neighbors from which the best remapping can be learned. To amend this problem, we modify the k-NN algorithm in the following way: instead of looking for neighbors among all the m' samples, we only consider a subset of $l \leqslant m'$ samples, where each value y only appears once. After the $\log(l)$ neighbors have been detected among the l samples, we select w according to a majority vote over the m' tuples (w, y) created through the remapping.

The distance on which the notion of neighbor is based depends on the experiments. We have considered the standard distance among numbers in the first and fourth experiments, the Euclidean distance in the second one, and the Manhattan distance between tuples of numbers in the third one (where the distance between the components is the standard numerical distance).

Concerning the ANN models, their specifics are in section 3.4. Note that, for the sake of fairness, we use the same architecture for both pre-processing methods, although we adapt

number of epochs and batch size to the particular dataset we are dealing with.

Further details relative to the specific experiments are provided the following sections, each of them discussing one of the above mentioned scenarios.

3.3.3 Frequentist approach

In the experiments, we will compare our method with the frequentist one. This approach has been proposed originally in [CCG10] for estimating mutual information, and extended successively also to min-entropy leakage [CKN13]. Although not considered in the literature, the extension to the case of g-vulnerability is straightforward. The method consists in estimating the probabilities that constitute the channel matrix C, and then calculating analytically the g-vulnerability on C. The precise definition is in section 3.4.

In [CCP19] it was observed that, at least in the case of the Bayes vulnerability[3], the frequentist approach performs poorly when the size of the observable domain $|\mathcal{Y}|$ is large with respect to the available data. We want to examine whether this is the case also for other vulnerabilities.

3.3.4 Experiment 1: multiple guesses

We consider a system in which the secrets $\mathcal{X}$ are the integers between 0 and 9, and the observables $\mathcal{Y}$ are the integers between 0 and 15999. Hence $|\mathcal{X}| = 10$ and $|\mathcal{Y}| = 16K$. The channel C adds noise to the elements of $\mathcal{X}$ according to the following geometric distribution:

$$C_{xy} = P_{Y|X}(y|x) = \lambda \, \exp(-\nu|r(x) - y|)\,, \tag{3.55}$$

where:

- ν is a parameter that determines how concentrated around $y = x$ the distribution is. In this experiment we set $\nu = 0.002$;
- r is just an auxiliary function that reports $\mathcal{X}$ to the same scale of $\mathcal{Y}$, and centers $\mathcal{X}$ on $\mathcal{Y}$. Here we have $r(x) = 1000\,x + 3499.5$;
- $\lambda = {e^{\nu}-1}/{(e^{\nu}+1)}$ is a normalization factor tuned so that eq. (3.55) is indeed a distribution.

fig. 3.1 illustrates the shape of C_{xy}. More precisely, it shows the distributions $P_{Y|X}(\cdot|x)$ for two adjacent secrets $x = 5$ and $x = 6$. We consider an adversary that can make two attempts to discover the secret (two-tries adversary), and we define the corresponding gain function as follows. A guess $w \in \mathcal{W}$ is one of all the possible combinations of 2 different secrets

[3] Strictly speaking, [CCP19] considered the estimation of the Bayes error, but the essence does not change since Bayes vulnerability and Bayes error are complementary with respect to 1.

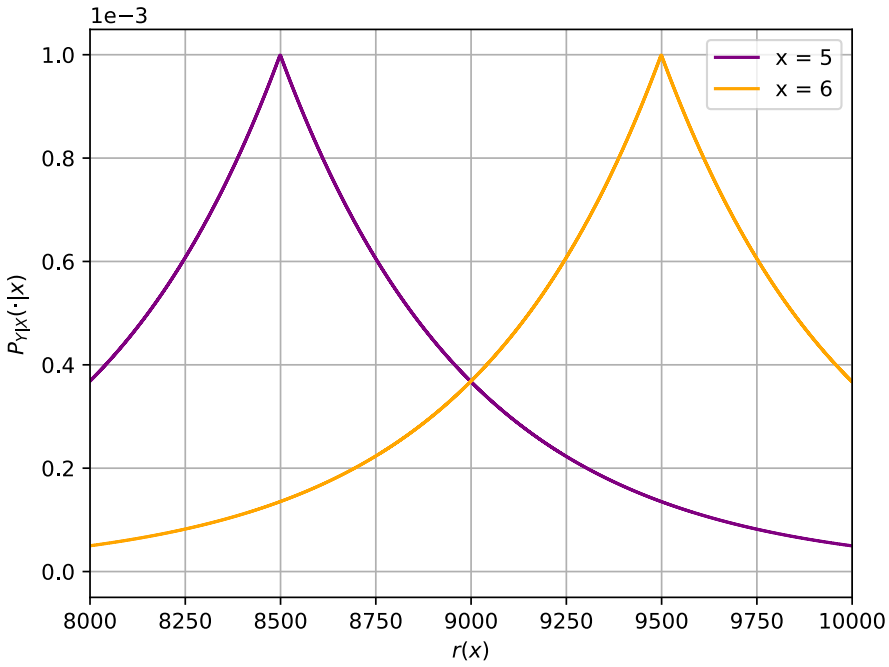


Figure 3.1 – The channel of section 3.3.4. The two curves represent the distributions $P_{Y|X}(\cdot|x)$ for two adjacent secrets: $x = 5$ and $x = 6$.

from $\mathcal{X}$, i.e., $w = \{x_0, x_1\}$ with $x_0, x_1 \in \mathcal{X}$ and $x_0 \neq x_1$. Therefore $|\mathcal{W}| = \binom{10}{2} = 45$. The gain function g is then

$$g(w, x) = \begin{cases} 1 & \text{if } x \in w \\ 0 & \text{otherwise}\,. \end{cases} \tag{3.56}$$

For this experiment we consider a uniform prior distribution π on $\mathcal{X}$. The true g-vulnerability for these particular ν and π, results to be $V_g = 0.892$. For all the following experiments in the multiple guesses scenario, the reported results are obtained evaluating the normalized estimation error on 50 different validation sets of 50K samples of type (x, y) drawn according to π and to the channel distribution in eq. (3.55). The sample sizes for the training sets (of the same type as $\mathcal{D}_m$) that we consider are 10K, 30K and 50K respectively. As explained before, for each size we use 5 different training sets where the samples are couples (y, x).

Data pre-processing

In this part of the experiment we transform the training data according to the data pre-processing described in section 3.2.1. The result are sets of samples of type (w, y) of the same type as $\mathcal{D}'_{m'}$. Then, we use the latter to learn the $\mathcal{Y} \to \mathcal{W}$ remapping, and we do so by using k-NN and ANN classifiers. The plot in fig. 3.2c shows the performances of the

k-NN and ANN models on the validation sets used to estimate the g-vulnerability in terms of normalized estimation error, while fig. 3.2f shows the same performances compared to those of the frequentist approach. As we can see, the precision of the frequentist is much lower, thus confirming that the trend observed by [CCP19] for the Bayes vulnerability holds also for other gain functions. Intuitively, this gap occurs especially when $|\mathcal{Y}|$ is high with respect to the number of training samples, and it is due to the fact that with the frequentist approach there is no real learning, so we cannot make a good guess with the observables never seen before. In ML on the contrary we can still make an informed guess, especially when the channel has a rather regular behavior, i.e., the noise expressed by the channel is "smooth" (cfr. [CCP19]).

It is worth noting that, in this experiment, the pre-processing of each sample (x, y) creates 9 samples (matching y with each possible $w \in \mathcal{W}$ such that $w = \{x, x'\}$ with $x' \neq x$). This means that the sample size of the pre-processed sets is 9 times the size of the original ones. For functions g representing more than 2 tries this pre-processing method may create training sets too large. In the next section we consider the alternative pre-processing method, showing that it can be a good compromise.

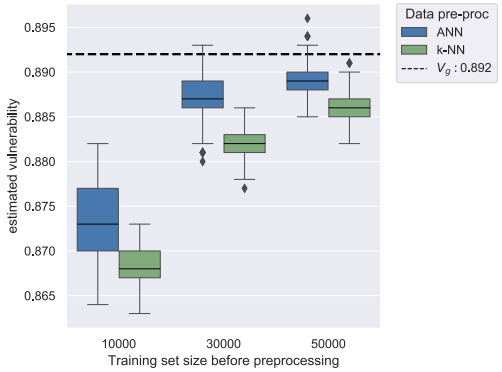
Channel pre-processing

Let us now suppose it is possible to interact with the channel modified according to the channel pre-processing method (cfr. section 3.2.2) so that we are able to sample data of the type (w, y) from it. With these samples we form training sets (of the same type as $\mathcal{D}''_{m''}$) of size 10K, 30K, and 50K. Then we proceed with the learning and the g-vulnerability estimation as before. The results are showed in fig. 3.2d. As we can see, the results are worse than in the data pre-processing case, especially for the k-NN algorithm. This was to be expected, since the random sampling to match the effect of g introduces a further level of confusion, as explained in section 3.2.2. Nevertheless, these results are still much better than the frequentist case, so it is a good alternative method to apply when the use of data pre-processing would generate validation sets that are too large, which could be the case when the matrix representing g contains large numbers with a small common divider.

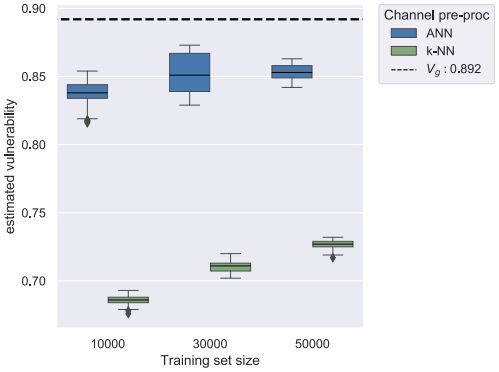
Figure 3.3 shows the estimation performances of all the three methods considering both pre-processing techniques. Table 3.2, table 3.3, table 3.4, and table 3.5 show the dispersion and total error values for both the data and the channel pre-processing methods. For the sake of simplicity, the values concerning the frequentist approach, which are not influenced by the pre-processing are identically reported for both cases. We indicate the size of the training set before the data pre-processing in table 3.2 and table 3.3.

3.3.5 Experiment 2: location privacy

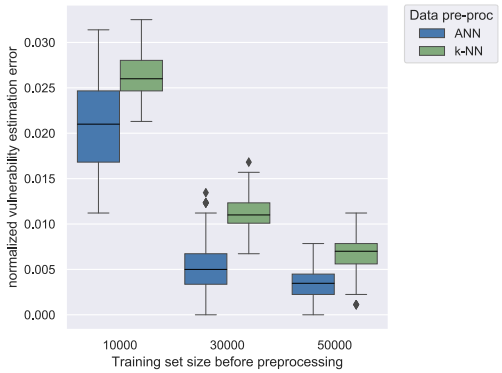
In this section we estimate the g-vulnerability of a typical system for location privacy protection. We use data from the open Gowalla dataset [LK], which contains the coordinates of users' check-ins. In particular, we consider a square region in San Francisco, USA, centered



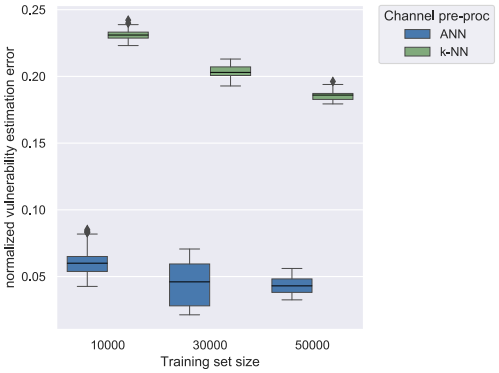
(a) Vulnerability estimation for ANN and k-NN with data pre-processing.



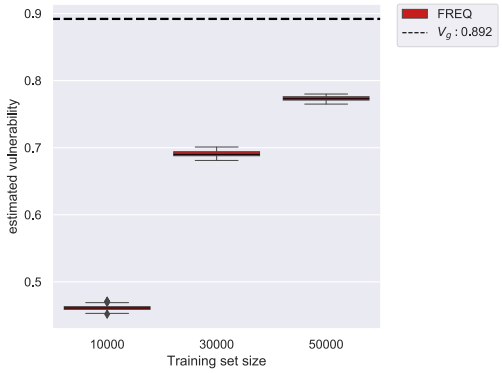
(b) Vulnerability estimation for ANN and k-NN with channel pre-processing.



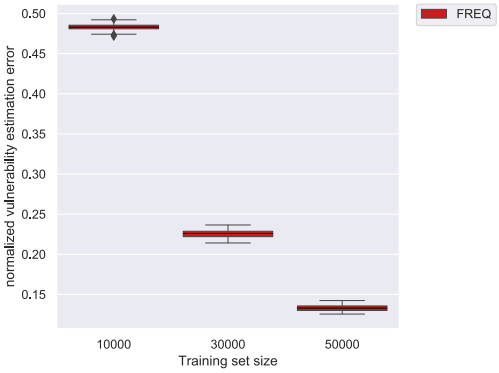
(c) Normalized estimation error for ANN and k-NN with data pre-processing.



(d) Normalized estimation error for ANN and k-NN with channel pre-processing.

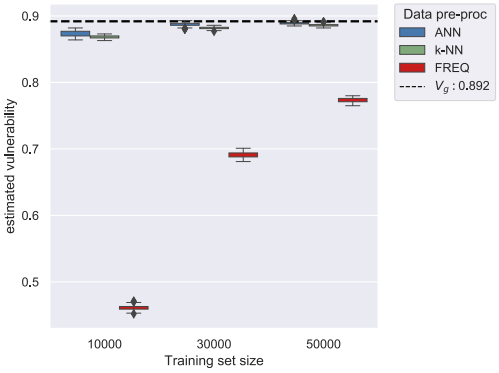


(e) Vulnerability estimation for the frequentist approach.

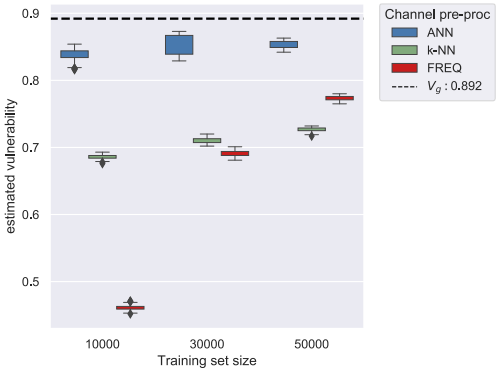


(f) Normalized estimation error for the frequentist approach.

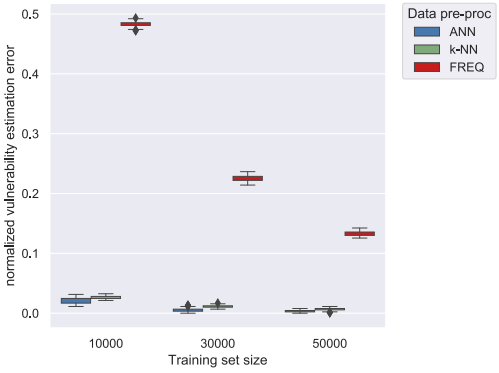
Figure 3.2 – Multiple guesses scenario: vulnerability estimation and normalized estimation error plots.



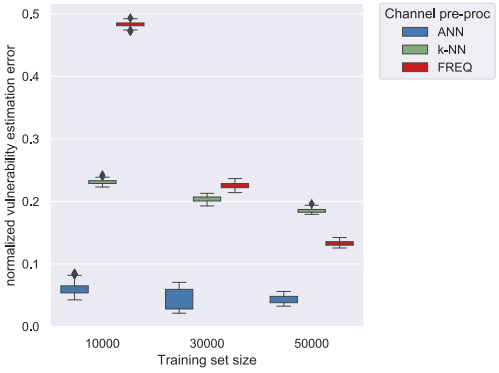
(a) Vulnerability estimation for ANN and k-NN with data pre-processing, and for the frequentist approach.



(b) Vulnerability estimation for ANN and k-NN with channel pre-processing, and for the frequentist approach.



(c) Normalized estimation error for ANN and k-NN with data pre-processing, and for the frequentist approach.



(d) Normalized estimation error for ANN and k-NN with channel pre-processing, and for the frequentist approach.

Figure 3.3 – Multiple guesses scenario: comparison plots.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
10000	0.005	0.002	0.004
30000	0.003	0.002	0.004
50000	0.002	0.002	0.003

Table 3.2 – Multiple guesses scenario, data pre-processing: dispersion.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
10000	0.022	0.026	0.483
30000	0.006	0.012	0.226
50000	0.004	0.007	0.133

Table 3.3 – Multiple guesses scenario, data pre-processing: total error.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
10000	0.012	0.003	0.004
30000	0.016	0.004	0.004
50000	0.006	0.003	0.003

Table 3.4 – Multiple guesses scenario, channel pre-processing: dispersion.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
10000	0.061	0.231	0.483
30000	0.048	0.203	0.226
50000	0.044	0.186	0.133

Table 3.5 – Multiple guesses scenario, channel pre-processing: total error.

in (latitude, longitude) = $(37.755, -122.440)$, and with 5Km long sides. In this area Gowalla contains 35162 check-ins.

We discretize the region in 400 cells of 250m long side, and we assume that the adversary's goal is to discover the cell in which a check-in is located. The frequency of the Gowalla check-ins per cell is represented by the heat-map in fig. 3.4. From these frequencies we can directly derive the distribution representing the prior of the secrets.

The channel C that we consider here is the optimal obfuscation mechanism proposed in [STT+12b] to protect location privacy under a utility constraint. We recall that the framework of [STT+12b] assumes two loss functions, one for utility and one for privacy. The utility loss of a mechanism, for a certain prior, is defined as the expected utility loss of the noisy data generated according to the prior and the mechanism. The privacy loss is defined in a similar way, except that we allow the attacker to "remap" the noisy data so to maximize the privacy loss. For our experiment, we use the Euclidean distance as loss function for the utility, and the g function defined in the next paragraph as loss function for the privacy. For further details on the construction of the optimal mechanism we refer to [STT+12b].

We define $\mathcal{X}, \mathcal{Y}$ and $\mathcal{W}$ to be the set of the cells. Hence $|\mathcal{X}| = |\mathcal{Y}| = |\mathcal{W}| = 400$. We consider a gain function that represents the precision of the guess in terms of euclidean distance: the idea is that the smaller is the distance between the real cell x and the guessed cell w, the higher is the gain. Specifically, our g is illustrated in fig. 3.5, where the central cell represents the real location x. For a generic "guess" cell w, the number written in w represent $g(w, x)$. Thus for example we have $g(x, x) = 4$, and $g(w, x) = 2$ if w is an immediate neighbor of x.[4]

[4]Formally, g is defined as $g(w, x) = \lfloor(\gamma \exp(-\alpha d(w, x)/l))\rceil$, where $\gamma = 4$ is the maximal gain, $\alpha = 0.95$ is a normalization coefficient to control the skewness of the exponential, d is the euclidean distance and $l = 250$

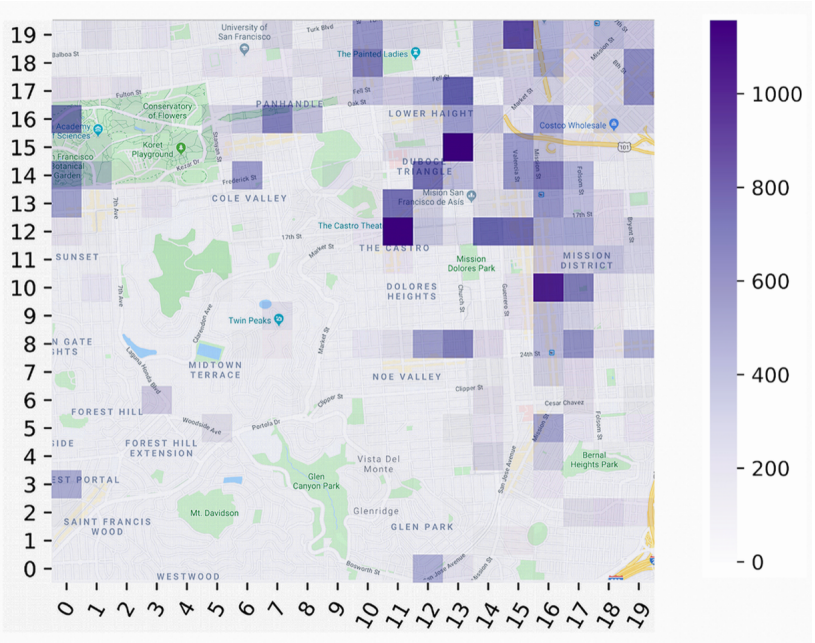


Figure 3.4 – Heat-map representing the Gowalla check-ins distribution in the area of interest; the density of check-ins in each cell is reported in the color bar on the side.

In this experiment we consider training set sizes of 100, 1k and 10K samples respectively.

After applying the data pre-processing transformation, the size of the resulting datasets is approximately 18 times that of the original one. This was to be expected, since the sum of the values of g in fig. 3.5 is 20. Note that this sum and the increase factor in the dataset do not necessarily coincide, because the latter is also influenced by the prior and by the mechanism.

Figure 3.6c and fig. 3.6d show the performance of k-NN and the ANN for the application of both data pre-processing and the channel pre-processing methods in terms of normalized estimation error. As expected, the data pre-processing method is more precise than the channel pre-processing one, although only slightly. The ANN model is also slightly better than the k-NN in most of the cases. For this experiment, as one can see in fig. 3.7, the frequentist approach outperforms the ML methods. Indeed this can be explained as follows: the observable space is not very large, which is a scenario where the frequentist approach can be successful because the available data is enough to estimate the real distribution. Indeed the ANN method performs quite well too, but the model we used is quite complex for this problem and therefore, we obtain a slightly worse performance than the frequentist approach. This is indeed an example which shows that the frequentist approach can still work well in certain scenarios. Although, as experimentally verified in all the other experiments, ML techniques, and in particular ANN, easily outperforms the frequentist approach when large

is the length of the cells' side. The symbol $\lfloor \cdot \rceil$ in this context represents the rounding to the closest integer operation.

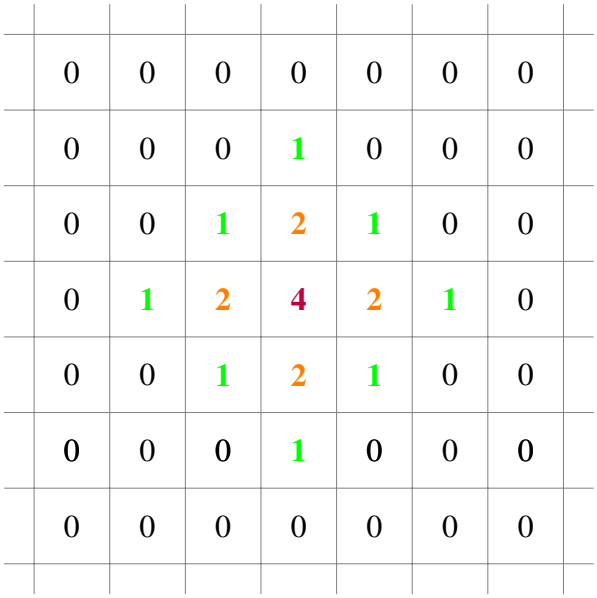


Figure 3.5 – The "diamond" shape created by the gain function around the real secret; the values represent the gains assigned to each guessed cell w when x is the central cell.

observable spaces are involved.

Table 3.6, table 3.7, table 3.8, and table 3.9 show the dispersion and total error values for both the data and the channel pre-processing methods.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
100	0.051	0.055	0.032
1000	0.031	0.014	0.008
10000	0.004	0.005	0.002

Table 3.6 – Location privacy scenario, data pre-processing: dispersion.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
100	0.379	0.357	0.318
1000	0.102	0.112	0.034
10000	0.024	0.053	0.004

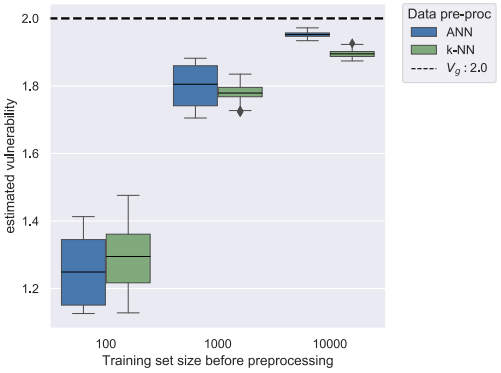
Table 3.7 – Location privacy scenario, data pre-processing: total error.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
100	0.033	0.038	0.032
1000	0.032	0.016	0.008
10000	0.013	0.015	0.002

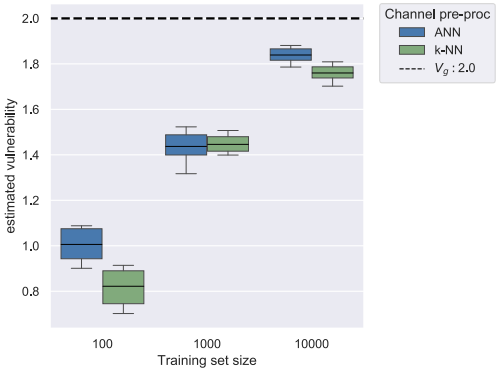
Table 3.8 – Location privacy scenario, channel pre-processing: dispersion.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
100	0.498	0.590	0.318
1000	0.283	0.278	0.034
10000	0.082	0.121	0.004

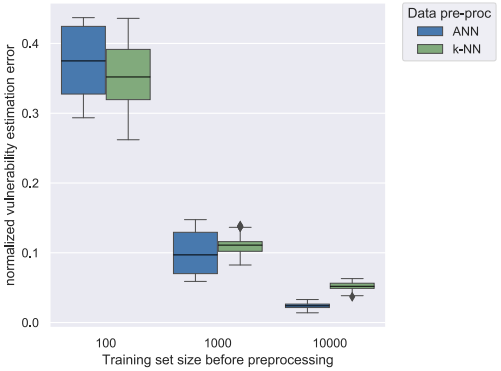
Table 3.9 – Location privacy scenario, channel pre-processing: total error.



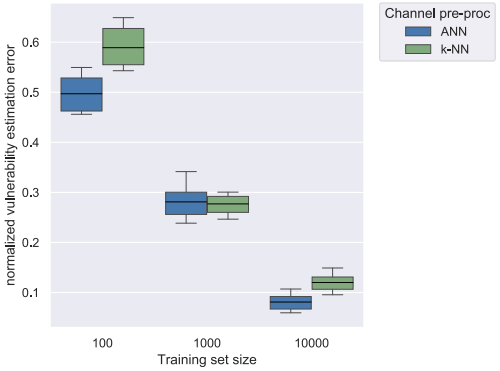
(a) Vulnerability estimation for ANN and k-NN with data pre-processing.



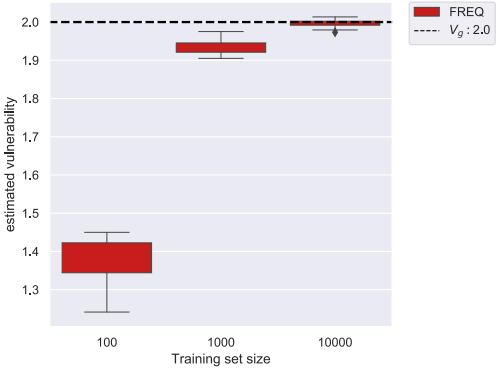
(b) Vulnerability estimation for ANN and k-NN with channel pre-processing.



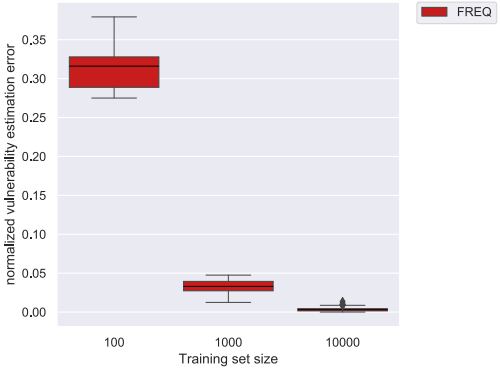
(c) Normalized estimation error for ANN and k-NN with data pre-processing.



(d) Normalized estimation error for ANN and k-NN with channel pre-processing.

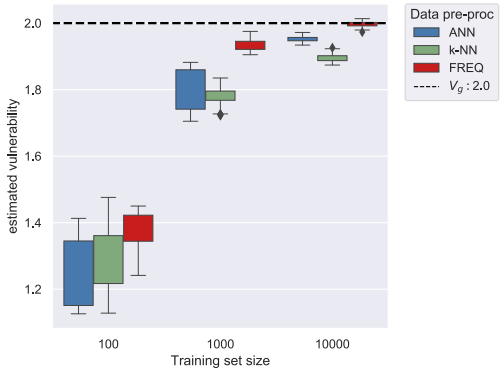


(e) Vulnerability estimation for the frequentist approach.

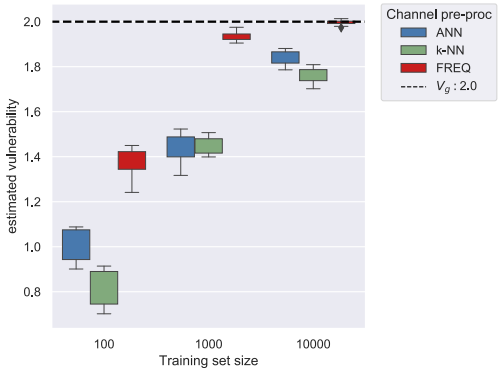


(f) Normalized estimation error for the frequentist approach.

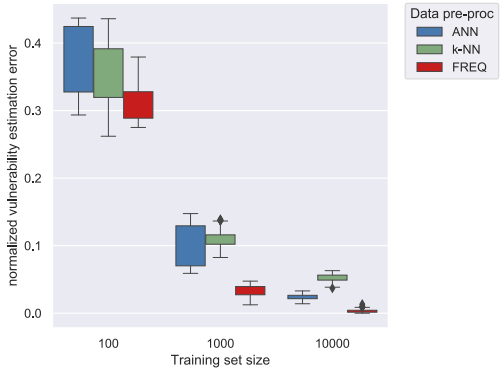
Figure 3.6 – Location privacy scenario: vulnerability estimation and normalized estimation error plots.



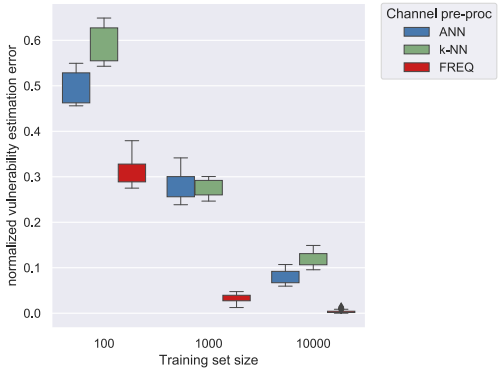
(a) Vulnerability estimation for ANN and k-NN with data pre-processing, and for the frequentist approach.



(b) Vulnerability estimation for ANN and k-NN with channel pre-processing, and for the frequentist approach.



(c) Normalized estimation error for ANN and k-NN with data pre-processing, and for the frequentist approach.



(d) Normalized estimation error for ANN and k-NN with channel pre-processing, and for the frequentist approach.

Figure 3.7 – Location privacy scenario: comparison plots.

3.3.6 Experiment 3: differential privacy

In this section we consider a popular application of DP: individual data protection in medical datasets from which we wish to extract some statistics via counting queries.

It is well known that the release of exact information from the database, even if it is only the result of statistical computation on the aggregated data, can leak sensitive information about the individuals. The solution proposed by DP is to obfuscate the released information with carefully crafted noise that obeys certain properties. The goal is to make it difficult to detect whether a certain individual is in the database or not. In other words, two adjacent datasets (i.e., datasets that differ only for the presence of one individual) should have almost the same likelihood to produce a certain observable result.

In our experiment, we consider the Cleveland heart disease dataset [DG17] which consist of 303 records of patients with a medical heart condition. Each condition is labeled by an integer number (label) that indicates how serious the disease is: from 0, which represents a healthy patient, to 4, which represents a patient whose life is at risk.

We assume that, to help medical research, the hospital releases the histogram of these labels, i.e., the counts of their occurrences in the database. We also assume that, for the sake of protecting their patients' privacy, the hospital sanitizes the histogram by adding geometric noise, which is a typical DP mechanism, to each label's count. More precisely, if the count of a label is z_1, the probability that the corresponding published number is z_2 is defined by the distribution in eq. (3.55), where x and y are replaced by z_1 and z_2 respectively, and r is 1. Note that z_1 is the real count, so it is an integer between 0 and 303, while its noisy version z_2 ranges on all integers. Concerning the value of ν, in this experiment it is set to 1.

The secrets space $\mathcal{X}$ is set to be a set of two elements: the full dataset, and the dataset with one record less. These are adjacent in the sense of DP, and, as customary in DP, we assume that the record on which the two databases differ is the target of the adversary. The observables space $\mathcal{Y}$ is the set of the 5-tuples produces by the noisy counts of the 5 labels. $\mathcal{W}$ is set to be the same as $\mathcal{X}$.

We assume that the adversary is interested especially in finding out whether the patient has a serious condition. The function g reflects this preference by assigning higher value to higher labels. Specifically, we set:

$$g(w,x) = \begin{cases} 0, & \text{if } w \neq x \\ 1, & \text{if } w = x \wedge x \in \{0,1,2\} \\ 2, & \text{if } w = x \wedge x \in \{3,4\}, \end{cases} \tag{3.57}$$

and $\mathcal{W} = \mathcal{X}$.

For the estimation, we consider 50 different validation sets of 50K samples each drawing them from the channel. The results reported below represent the estimation performance of each model on these validation sets.

As training set sizes, we consider 10K, 30K and 50K samples, that we use to learn the remapping from the obfuscated counts, $\mathcal{Y}$, to the guesses about the diseases $\mathcal{W}$. For each of these sizes we consider 5 training sets. Again, as ML algorithms we use ANN and k-NN.

For the latter we have to decide what kind of distance we use for evaluating the proximity of the elements of $\mathcal{Y}$. We choose the Manhattan distance, which seems the most natural in this case[5] In the case of data pre-processing the size of the transformed training sets (of the same type as $\mathcal{D}'_{m'}$) is about 1.2 times the size of the original training sets (of the same type as $\mathcal{D}_m$). The performance is shown in fig. 3.8c and fig. 3.8a. For the channel pre-processing approach, the performance is shown in fig. 3.8d and fig. 3.8b. Surprisingly, in this case the data pre-processing method outperforms the channel pre-processing one, although only slightly. Additional plots, including the comparison to the frequentist approach, can be found in fig. 3.9.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
10000	0.032	0.006	0.004
30000	0.011	0.005	0.005
50000	0.010	0.004	0.004

Table 3.10 – Differential privacy scenario, data pre-processing: dispersion.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
10000	0.062	0.045	0.145
30000	0.021	0.030	0.104
50000	0.016	0.024	0.086

Table 3.11 – Differential privacy scenario, data pre-processing: total error.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
10000	0.025	0.006	0.004
30000	0.008	0.004	0.005
50000	0.003	0.005	0.004

Table 3.12 – Differential privacy scenario, channel pre-processing: dispersion.

	Estimation method		
Tr. set size	ANN	k-NN	FREQ
10000	0.059	0.040	0.145
30000	0.012	0.027	0.104
50000	0.005	0.022	0.086

Table 3.13 – Differential privacy scenario, channel pre-processing: total error.

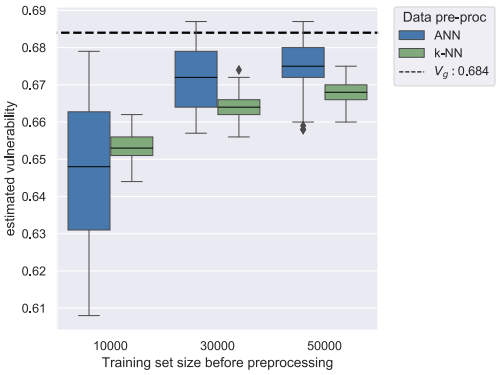
Table 3.10, table 3.11, table 3.12, and table 3.13 show the the dispersion and total error values for both the data and the channel pre-processing methods.

3.3.7 Experiment 4: password checker

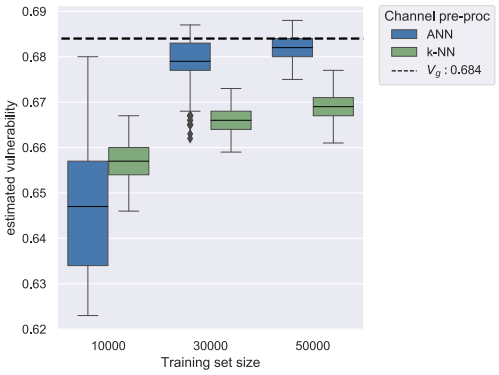
Table 3.14, table 3.15, represent the dispersion and the total error for section 3.3.7.

In this experiment we consider a password checker, namely a program that tests whether a given string corresponds to the password stored in the system. We assume that string and password are sequences of 128 bits, an that the program is "leaky", in the sense that it checks the two sequences bit by bit and it stops checking as soon as it finds a mismatch, reporting failure. It is well known that this opens the way to a timing attack (a kind of side-channel attack), so we assume that the system tries to mitigate the threat by adding some random

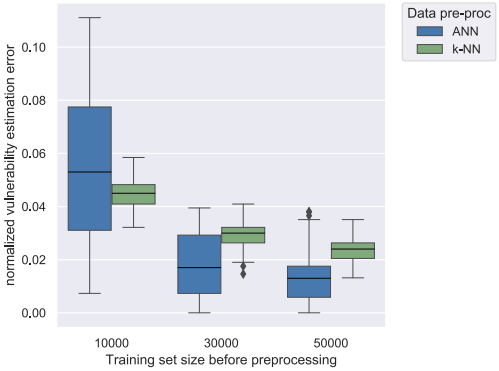
[5]The Manhattan distance on histograms corresponds to the total variation distance on the distributions resulting from the normalization of these histograms.



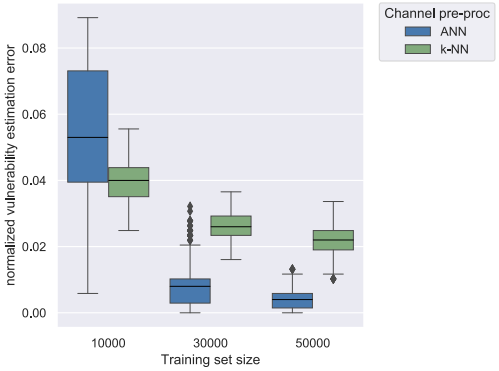
(a) Vulnerability estimation for ANN and k-NN with data pre-processing.



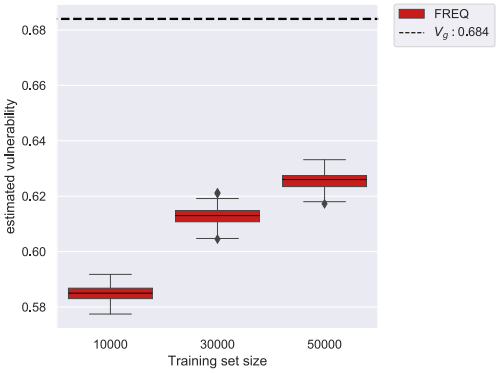
(b) Vulnerability estimation for ANN and k-NN with channel pre-processing.



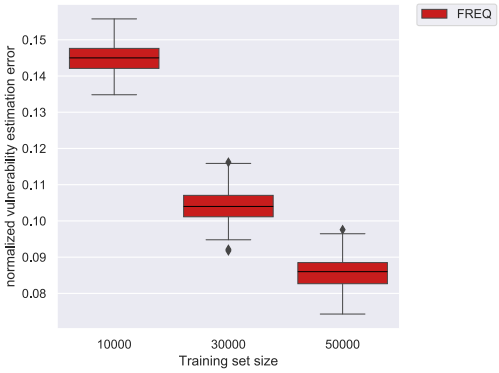
(c) Normalized estimation error for ANN and k-NN with data pre-processing.



(d) Normalized estimation error for ANN and k-NN with channel pre-processing.

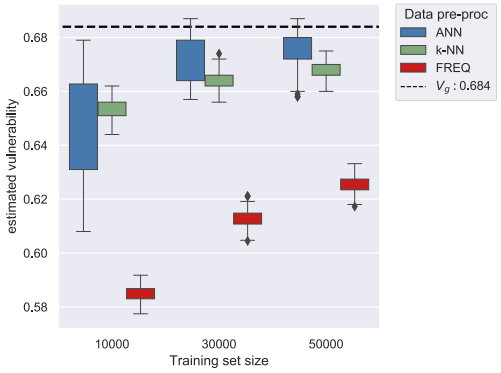


(e) Vulnerability estimation for the frequentist approach.

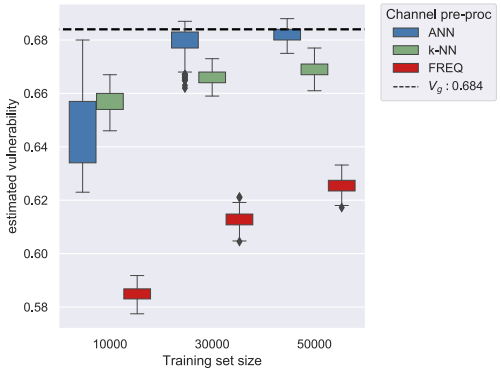


(f) Normalized estimation error for the frequentist approach.

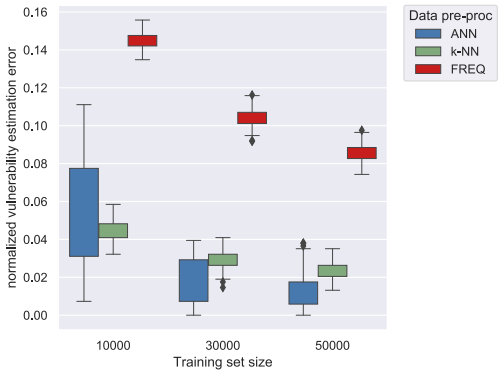
Figure 3.8 – Differential privacy scenario: vulnerability estimation and normalized estimation error plots.



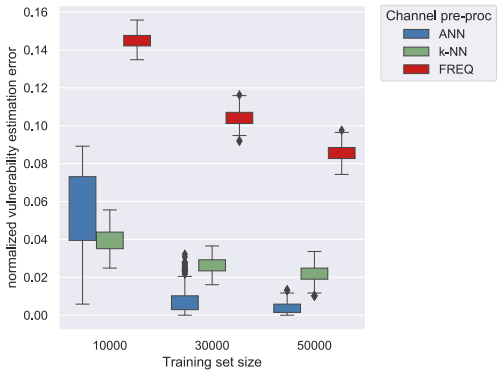
(a) Vulnerability estimation for ANN and k-NN with data pre-processing, and for the frequentist approach.



(b) Vulnerability estimation for ANN and k-NN with channel pre-processing, and for the frequentist approach.



(c) Normalized estimation error for ANN and k-NN with data pre-processing, and for the frequentist approach.



(d) Normalized estimation error for ANN and k-NN with channel pre-processing, and for the frequentist approach.

Figure 3.9 – Differential privacy scenario: comparison plots.

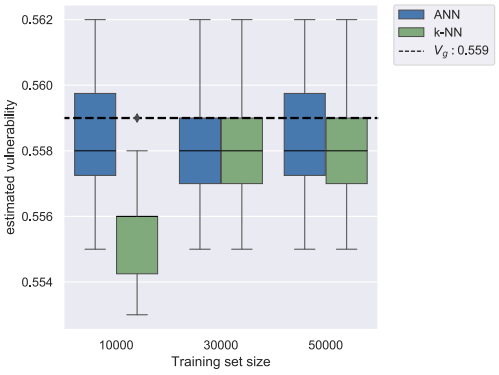
	Estimation method		
Tr. set size	ANN	k-NN	FREQ
10000	0.002	0.003	0.004
30000	0.002	0.002	0.003
50000	0.002	0.002	0.003

Table 3.14 – Password checker scenario: dispersion.

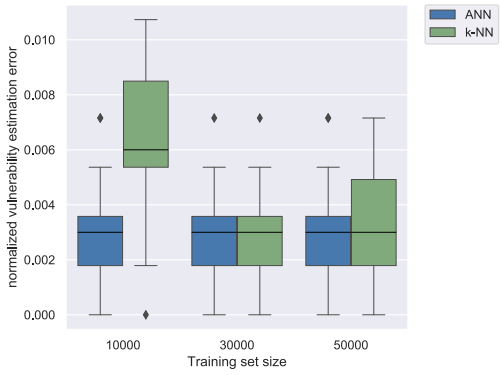
	Estimation method		
Tr. set size	ANN	k-NN	FREQ
10000	0.003	0.007	0.015
30000	0.003	0.004	0.007
50000	0.003	0.004	0.005

Table 3.15 – Password checker scenario: total error.

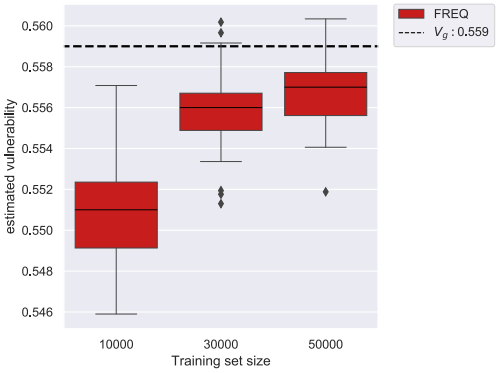
delay, sampled from a Laplace distribution and then bucketing the reported time in 128 bins corresponding to the positions in the sequence (or equivalently, by sampling the delay from a Geometric distribution, cfr. eq. (3.55)). Hence the channel C is a $2^{128} \times 128$ stochastic matrix.



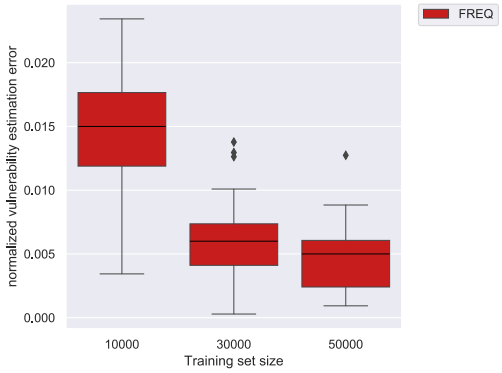
(a) Vulnerability estimation for ANN and k-NN.



(b) Normalized estimation error for for ANN and k-NN.



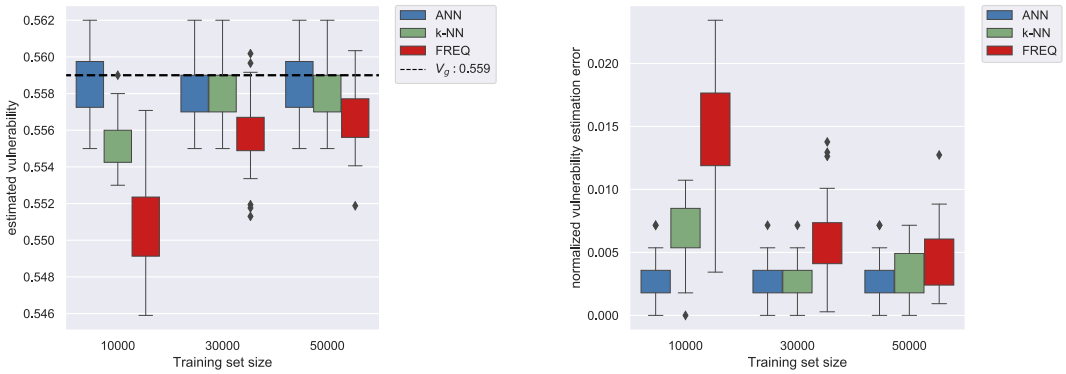
(c) Vulnerability estimation for the frequentist approach.



(d) Normalized estimation error for the frequentist approach.

Figure 3.10 – Password checker scenario: vulnerability estimation and normalized estimation error plots.

The typical attacker is an interactive one, which figures out larger and larger prefixes of the password by testing each bit at a time. We assume that the attacker has already figured out the first 6 bits of the sequence and it is trying to figure out the 7-th. Thus the prior π is distributed (uniformly, we assume) only on the sequences formed by the known 6-bits prefix and all the possible remaining 122 bits, while the g function assigns 1 to the sequences whose 7-th bit agrees with the stored password, and 0 otherwise. Thus g is a *partition gain function* [ACPS12], and its particularity is that for such kind of functions data pre-processing and channel pre-processing coincide. This is because $g(w, x)$ is either 0 or 1, so in both cases we generate exactly one pair (w, y) for each pair (x, y) for which $g(w, x) = 1$. Note that in this case the data pre-processing transformation does not increase the training set, and the channel pre-processing transformation does not introduce any additional noise. The RC matrix (cfr. section 3.2.1) is a 2×128 stochastic matrix.



(a) Vulnerability estimation for ANN, k-NN, and the frequentist approach.

(b) Normalized estimation error for ANN, k-NN, and the frequentist approach.

Figure 3.11 – Password checker scenario: comparison plots.

The experiments are done with training sets of 10K, 30K and 50K samples. The results are reported in fig. 3.10. We note that the estimation error is quite small, especially in the ANN case. This is because the learning problem is particularly simple since, by considering the g-leakage and the preprocessing, we have managed to reduce the problem to learning a function of type $\mathcal{Y} \to \mathcal{W}$, rather than $\mathcal{Y} \to \mathcal{X}$, and there is a huge difference in size between $\mathcal{W}$ and $\mathcal{X}$ (the first is 2 and the latter is 2^{128}). Also the frequentist approach does quite well, and this is because $\mathcal{Y}$ is small. With a finer bucketing (on top of the Laplace delay), or no bucketing at all, we expect that the difference between the accuracy of the frequentist and of the ML estimation would be much larger.

Note that with a non-leaky password checker the observables are only $fail$ or $success$. In this case the size of $\mathcal{Y}$ would be 2, but since *success* would have an extremely small probability ($1/2^{128}$), the vulnerability would be negligible, and it would not be detected neither by our approach nor by the frequentist one, because a pair $(\cdot, success)$ would never be generated in practice. Hence both approaches would report vulnerability 0.

3.4 Technical details

In this section we report a few technical details concerning the ANN models' parameters and the frequentist approach.

ANN parameters details

In table 3.16 we report the values assigned to the hyper-parameters for the models used in the experiments above. We list here the specifics for the ANNs models used in the experiments. All the models are simple feed-forward networks whose layers are fully connected. The activation functions for the hidden neurons are rectifier linear functions, while the output layer has softmax activation function.

The loss function minimized during the training is the cross entropy, a popular choice in classification problems. The remapping $\mathcal{Y} \to \mathcal{W}$ can be in fact considered as a classification problem such that, given an observable, a model learns to make the best guess.

For each experiments, the models have been tuned by cross-validating them using one randomly chosen training sets among the available ones choosing among the largest in terms of samples.

		Hyper-parameters				
Experiment	**Pre-processing**	learning rate	hidden layers	epochs	hidden units per layer	batch size
Multiple guesses	Data	10^{-3}	3	700	$[100, 100, 100]$	1000
	Channel	10^{-3}	3	500	$[100, 100, 100]$	1000
Location Priv.	Data	10^{-3}	3	1000	$[500, 500, 500]$	$200, 500, 1000$
	Channel	10^{-3}	3	$200, 500, 1000$	$[500, 500, 500]$	$20, 200, 500$
Diff. Priv.	Data	10^{-3}	3	500	$[100, 100, 100]$	200
	Channel	10^{-3}	3	500	$[100, 100, 100]$	200
Psw SCA	-	10^{-3}	3	700	$[100, 100, 100]$	1000

Table 3.16 – Table with the hyper-parameters setting for each one of the experiments above. When multiple values are provided for the parameters of an experiment it is to be intended that each value corresponds to a specific size of the training set (sorted from the smallest to the largest number of samples).

Frequentist approach description

In the frequentist approach the elements of the channel, namely the conditional probabilities $P_{Y|X}(y|x)$, are estimated directly in the following way: the empirical prior probability of x, $\widehat{\pi}_x$, is computed by counting the number of occurrences of x in the training set and dividing the result by the total number of elements. Analogously, the empirical joint probability $\widehat{P}_{XY}(x, y)$ is computed by counting the number of occurrences of the pair (x, y) and dividing the result by the total number of elements in the set. The estimation $\widehat{C}_{xy}$ of C_{xy} is then defined as

$$\widehat{C}_{xy} = \frac{\widehat{P}_{XY}(x, y)}{\widehat{\pi}(x)}. \tag{3.58}$$

In order to have a fair comparison with our approach, which takes advantage of the fact that we have several training sets and validation sets at our disposal, while preserving at the same time the spirit of the frequentist approach, we proceed as follows: Let us consider a training set $\mathcal{D}_m$, that we will use to learn the best remapping $\mathcal{Y} \to \mathcal{W}$, and a validation set $\mathcal{T}_n$ which is then used to actually estimate the g-vulnerability. We first compute $\widehat{\pi}$ using $\mathcal{D}_m$. For each y in $\mathcal{Y}$ and for each $x \in \mathcal{X}$, the empirical probability $\widehat{P}_{X|Y}$ is computed using $\mathcal{D}_m$ as well. In particular, $\widehat{P}_{X|Y}(x|y)$ is given by the number of times x appears in pair with y divided by the number of occurrences of y. In case a certain y is in $\mathcal{T}_n$ but not in $\mathcal{D}_m$, it is assigned the secret $x' = \operatorname{argmax}_{x\in\mathcal{X}} \widehat{\pi}$ so that $\widehat{P}_{X|Y}(x'|y) = 1$ and $\widehat{P}_{X|Y}(x) = 0, \forall x \neq x'$. It is now possible to find the best mapping for each y defined as $w(y) = \operatorname{argmax}_{w\in\mathcal{W}} \sum_{x\in\mathcal{X}} \widehat{P}_{X|Y}(x|y)g(w,x)$. Now we compute the empirical joint distribution for each (x, y) in $\mathcal{T}_n$, namely $\widehat{Q}_{XY}$, as the number of occurrences of (x, y) divided by the total number of samples in $\mathcal{T}_n$. We now estimate the g-vulnerability on the validation samples according to:

$$\widehat{V}_n = \sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} \widehat{Q}_{XY}(x,y)g(w(y),x). \tag{3.59}$$

3.5 Final remarks

Our contribution

In this chapter we have presented a method to estimate the g-vulnerability in the black-box scenario and relying on the generalization capabilities of ML models. By estimating the g-vulnerability we can evaluate the leakage when a given system is attacked by many adversaries whose targets are different.

We provide a study of the learnability problem for the statistical point of view. We provide statistical guarantees showing the learnability of the g-vulnerability for all distributions and we derive distribution-free bounds on the accuracy of its estimation.

Finally we validate the performance of our method via several experiments using k-NN and ANN models. The code to run these experiments is available at the URL https://github.com/LEAVESrepo/leaves.

Related work

Most of the literature related to what has been discussed in this chapter has been presented in section 1.2. However, here, we would like to draw the reader's attention to a work which has been of fundamental importance in the study of the channel pre-processing approach, i.e. [BS16]. In this work, the authors investigated the indirect leakage induced by a channel (i.e., leakage on sensitive information not in the domain of the channel), and proved a fundamental equivalence between Dalenius min-entropy leakage under arbitrary correlations and g-leakage under arbitrary gain functions. This result is similar to our Theorem 3,

and it opens the way to the possible extension of our approach to this more general leakage scenario.

Summary

In this chapter, we have shown how ML can be used to model attacks and then study the amount of information leaked by the system whose outputs can be observed by the attackers. We have proposed the estimation of g-leakage for its flexibility and power to encompass many different attacks.

In particular, we have introduced two techniques, the data and channel pre-processing respectively, to reduce the problem of the g-vulnerability estimation to that of the Bayes risk estimation.

Since for each (x, y) all the possible correspondences (w, y) are created according to the data pre-processing, in regimes where the amount of samples collected from the system is small, it tends to outperform the channel pre-processing. However, in case the gain function g take very high values, the data pre-processing might quickly create very large large sets of samples. Moreover if the gain function takes non integer values, then for the data pre-processing some approximations are needed as explained above.

In the next chapter we will investigate and discuss a ML based method to protect the information via controlled noise injection in the original information input in the system.

CHAPTER 4

Privacy protection mechanisms design via machine learning

In this section we thoroughly discuss our ML based framework that can be used to build and deploy privacy protection mechanisms as anticipated in section 1.3. We focus on mechanisms that obfuscate the data by adding controlled noise. Usually the QoS that the user receives in exchange for his obfuscated data degrades with the amount of obfuscation, hence the challenge is to find a good trade-off between privacy and utility. Following the approach of [STT17], we aim at maximizing the privacy protection while preserving the desired QoS[1]. The framework that we develop is general and can be applied to any situation in which an attacker might infer sensitive information from accessible data correlated with such information. As a case study, we consider the problem of *location privacy* and in particular the *re-identification* of the user from her/his location. In the location privacy scenario, utility is typically expressed as a bound on the expected distance between the real location and the obfuscated one[2] [STT17, ABCP13, BCP14, CEP17], capturing the fact that location based services usually offer a better QoS when they receive a more accurate location.

As already mentioned in the introduction, if both privacy and utility can be expressed as a linear function, then the optimal trade-off can in principle be achieved with linear programming [STT17, BCP14, Sho15, OTPG17]. The limitation of this approach, however, is that it does not scale to large datasets. The problem is that the linear program needs one variable for every pair (w, z) of real and obfuscated locations. Such variables represent the probability of producing the obfuscated location z when the real one is w. For a 50×50 grid this is more than six million variables, which is already at the limit of what modern solvers can do. For a 260×260 grid, the program has 4.5 billion variables, making it completely intractable (we could not even launch such a program due to the huge memory requirements). Furthermore, the background knowledge and the correlation between data points affect privacy and are usually difficult to determine and express formally. Another

[1] Other approaches take the opposite view, and aim at maximizing utility while achieving the desired amount of privacy, see for instance [BCP14].

[2] This notion is known as *distortion* in information theory [CT06].

way to deal with the problem of finding a good trade-off between privacy and utility is using analytic methods (such as the Planar Laplace mechanism), which is the typical setting in the field of differential privacy.

We compare our proposed framework to both the linear programming based solution (when the computation is feasible) and to the Laplace mechanism. From table 4.1, which anticipates some of the experimental results in section 4.4.1 and section 4.5, we can see that our mechanism compares the optimal solution both when the synthetic and real data from the Gowalla dataset [LK] are involved. Likewise, we can asses that, in terms of Bayes error, the solution we propose outperforms the Laplace mechanism. However, for the sake of fairness, we must highlight that the planar Laplace was designed to satisfies a different notion of privacy, called *geo-indistinguishability* [ABCP13] (see next paragraph). Our mechanism on the contrary does not satisfy this notion. The main difference between the two approaches is that the scenarios where the planar Laplace is involved are concerned worst-case measures, i.e. with the protection of *each* individual datum. Conversely, as we will explain in the following sections, the mutual information and Bayes risk notions in our approach are average, in the sense that they are concerned with the *expected* level of privacy over all sensitive data. Clearly, the the former is a stronger notion of privacy, as proved in [AACP11] and [De12]. Its relation with mutual information has been explored in [Mir13, CY16], while its relation with the Bayes vulnerability has been investigated in [AAC+15]. In [CY16] it has been has proved that a conditional version of mutual information correspond to a relaxed form of differential privacy called (ε, δ)-differential privacy. A natural development of the current proposed framework will involve studying the possibility of generating worst-case mechanisms via ML in future work.

From a practical point of view our method belongs to the *local privacy* category, like LDP and geo-indistinguishability, in the sense that it can be deployed at the user's end, with no need of a trusted third party. Once the training is done the system can be used as a personal device that, each time the user needs to report his location to a LBS, generates a sanitized version of it by adding noise to the real location.

Synthetic data, low utility		
Laplace	Ours	Optimal
0.39	0.74	0.75

Synthetic data, high utility		
Laplace	Ours	Optimal
0.23	0.42	0.50

Gowalla data, low utility		
Laplace	Ours	Optimal
0.33	0.80	0.83

Gowalla data, high utility		
Laplace	Ours	Optimal
0.28	0.38	?

Table 4.1 – Bayes error on synthetic and Gowalla data, for the Laplace mechanism, our mechanism, and the optimal one, on a grid of 260×260 cells. In the last table the Bayes error of the optimal mechanism is unknown: the linear program contains 4.5 billion variables, making it intractable in practice.

4.1 Game theoretic problem description

Inspired by the GAN paradigm [GPAM+14], we propose a system consisting of two adversarial neural networks, G (*generator*) and C (*classifier*). The idea is that G generates noise so to confuse the adversary as much as possible, within the boundaries of the utility constraints, while C inputs the noisy locations produced by G and tries to re-identify (classify) the corresponding user. While fighting against C, G refines its strategy, until a point where it cannot improve any longer. Note that a significant difference from the standard GAN is that, in the latter, the generator has to learn to reproduce an existing distribution from samples. In our case, instead, the generator has to "invent" a distribution from scratch.

The interplay between G and C can be seen as an instance of a zero-sum Stackelberg game [STT17], where G is the *leader*, and C is the *follower*, and the payoff function f is the privacy loss. Finding the optimal point of equilibrium between G and C corresponds to solving a minimax problem on f with G being the minimizer and C the maximizer.

A major challenge in our setting is represented by the choice of f. A first idea would be to measure it in terms of C's capability to re-associate a location to the right user. Hence we could define f as the expected success probability of C's classification.

Such function f would be convex/concave with respect to the strategies of G and C respectively, so from game theory we would derive the existence of a saddle point corresponding to the optimal obfuscation-re-identification pair. The problem, however, is that it is difficult to reach the saddle point via the typical alternation between the two nets. Let us clarify this point by providing the following simple example[3]:

Example 1: Consider two users, Alice and Bob, in locations a and b respectively. Assume that at first G reports their true locations (no noise). Then C learns that a corresponds to $Alice$ and b to Bob. At the next round, G will figure that to maximize the misclassification error (given the prediction of C) it should swap the locations, i.e., report a for $Alice$ and b for Bob. Then, on its turn, C will have to "unlearn" the previous classification and learn the new one. But then, at the next round, G will again swap the locations, and bring the situation back to the starting point, and so on, without ever reaching an equilibrium. Note that a possible equilibrium point for G would be the mixed strategy that reports a for both $Alice$ and Bob[4] (so that C could only make a bling guess), but G may not stop there. The problem is that it is difficult to calibrate the training of G so that it stops in proximity of the saddle point rather than continuing all the way to reach its relative optimum. The situation is illustrated in table 4.2a.

In order to address this issue we adopt a different target function, less sensitive to the particular labeling strategy of C. The idea is to consider not just the *precision* of the classification, but, rather, the *information* contained in it.

There are two main ways of formalizing this intuition: the *mutual information* $I(X;Y)$

[3] A similar example was independently pointed out in [AA16].

[4] There are two more equilibrium points: one is when both $Alice$ and Bob report a or b with uniform probability, the other is when they both report b. All the three strategies are equivalent.

		C			
		$a\to A$ $b\to B$	$a\to A$ $b\to A$	$a\to B$ $b\to B$	$a\to B$ $b\to A$
G	$A\to a$ $B\to b$	1	0.5	0.5	0
	...	...	...	...	...
	$A\to a$ $B\to a$	0.5	0.5	0.5	0.5
	...	...	...	...	...
	$A\to b$ $B\to a$	0	0.5	0.5	1

(a) f = Expected success probability of the classification.

		C			
		$a\to A$ $b\to B$	$a\to A$ $b\to A$	$a\to B$ $b\to B$	$a\to B$ $b\to A$
G	$A\to a$ $B\to b$	**1** 𝟙	**0** 0.5	**0** 0.5	**1** 𝟙
	...	...	...	...	...
	$A\to a$ $B\to a$	**0** 0.5	**0** 0.5	**0** 0.5	**0** 0.5
	...	...	...	...	...
	$A\to b$ $B\to a$	**1** 𝟙	**0** 0.5	**0** 0.5	**1** 𝟙

(b) **Bold:** $f = I(X;Y)$. Hollow: $f = 1 - B(X|Y)$.

Table 4.2 – Payoff tables of the games in example 1, for various payoff functions f. A stands for $Alice$ and B for Bob.

and the *Bayes error* $B(X|Y)$, where X, Y are respectively the random variable associated to the *true ids*, and to the ids resulting from the classification (*predicted ids*). We recall that $I(X;Y) = H(X) - H(X|Y)$, where $H(X)$ is the entropy of X and $H(X|Y)$ is the residual entropy of X given Y, while $B(X|Y)$ is the probability of error when we select the value of X with *maximum aposteriori probability*, given Y.

Mutual information and Bayes error are related by the Santhi-Vardy bound [SV06]: $B(X|Y) \leqslant 1 - 2^{-H(X|Y)}$. If we set f to be $I(X;Y)$ or $1 - B(X|Y)$, we obtain the payoff table illustrated in table 4.2b. Note that the mimimum f in the first and last columns corresponds now to a point of equilibrium for any choice of C. This is not always the case, but in general it is closer to the equilibrium and makes the training of G more stable: training G for a longer time does not risk to increase the distance from the equilibrium point.

In this chapter we introduce the use the mutual information to generate the noise, but we evaluate the level of privacy also in terms of the Bayes error, which represents the probability of error of the strongest possible adversary. Both notions have been used in the literature as privacy measures, for instance mutual information has been applied to quantify anonymity [ZB05, CPP08b]. The Bayes error has been considered in [CPP08b, MMM10,

Che17, ACM+16], and indirectly as *min-entropy leakage* in [Smi09]. In [OTPG17], the authors claim that to guarantee a good level of location privacy a mechanism should measure well in terms of both the Bayes error and the residual entropy (which is strictly related to mutual information).

4.2 Our setting

We formulate the privacy-utility optimization problem using a framework similar to that of [BWI16]. We consider four random variables, X, Y, Z, W, ranging over the sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ and $\mathcal{W}$ respectively, with the following meaning:

- X: the sensitive information that the users wishes to conceal,
- W: the useful information with respect to some service provider and the intended notion of utility,
- Z: the information made visible to the service provider, which may be intercepted by some attacker, and
- Y: the information inferred by the attacker.

We assume a fixed joint distribution (*data model*) $P_{X,W}$ over the users' data $\mathcal{X} \times \mathcal{W}$. We present our framework assuming that the variables are discrete, but all results and definitions can be transferred to the continuous case, by replacing the distributions with probability density functions, and the summations with integrals. For the initial definitions and results of this section $\mathcal{X}$ and $\mathcal{Y}$ may be different sets. Starting from section 4.3 we will assume that $\mathcal{X} = \mathcal{Y}$.

An obfuscation mechanism can be represented as a conditional probability distribution $P_{Z|W}$, where $P_{Z|W}(z|w)$ indicates the probability that the mechanism transform the data point w into the noisy data point z. We assume that Z are the only attributes visible to the attacker and to the service provider. The goal of the defender G is to optimize the data release mechanism $P_{Z|W}$ so to achieve a desired level of utility while minimizing the leakage of the sensitive attributes X. The goal of the attacker C is to retrieve X from Z as precisely as possible. In doing so, it produces a classification $P_{Y|Z}$ (*prediction*).

Note that the four random variables form a Markov chain:

$$X \leftrightarrow W \leftrightarrow Z \leftrightarrow Y. \tag{4.1}$$

Their joint distribution is completely determined by the data model, the obfuscation mechanism and the classification:

$$P_{X,W,Z,Y}(x, w, z, y) \;=\; P_{X,W}(x, w) P_{Z|W}(z \mid w) P_{Y|Z}(y \mid z).$$

Symbol	Description
C	Classifier network (attacker).
G	Generator network.
$X, \mathcal{X}$	Sensitive information. (Random var. and domain.)
$W, \mathcal{W}$	Useful information with respect to the intended notion of utility.
$Z, \mathcal{Z}$	Obfuscated information accessible to the service provider and to the attacker.
$Y, \mathcal{Y}$	Information inferred by the attacker.
$P_{\cdot,\cdot}$	Joint probability of two random variables.
$P_{\cdot\|\cdot}$	Conditional probability.
$P_{Z\|W}$	Obfuscation mechanism.
$B(\cdot \mid \cdot)$	Bayes error.
$\mathbb{L}[Z \mid W]$	Utility loss induced by the obfuscation mechanism.
L	Threshold on the utility loss.
$H(\cdot)$	Entropy of a random variable.
$H(\cdot\|\cdot)$	Conditional entropy.
$I(\cdot;\cdot)$	Mutual information between two random variables.

Table 4.3 – Table of symbols for chapter 4

From $P_{X,W,Z,Y}$ we can derive the marginals, the conditional probabilities of any two variables, etc. For instance:

$$P_X(x) = \sum_w P_{X,W}(x, w). \tag{4.2}$$

$$P_Z(z) = \sum_{xw} P_{X,W}(x, w) P_{Z|W}(z \mid w). \tag{4.3}$$

$$P_{Z|X}(z|x) = \frac{\sum_w P_{X,W}(x, w) P_{Z|W}(z \mid w)}{P_X(x)}. \tag{4.4}$$

$$P_{X|Z}(x|z) = \frac{P_{Z|X}(z|x) P_X(x)}{P_Z(z)}. \tag{4.5}$$

The latter distribution, $P_{X|Z}$, is the *posterior distribution* of X given Z, and plays an important role in the following sections.

4.2.1 Quantifying utility

Concerning the utility, we consider a loss function $\ell : W \times Z \to [0, \infty)$, where $\ell(w, z)$ represents the utility loss caused by reporting z when the true value is w.

Definition 2 (Utility loss)**:** The utility loss from the original data W to the noisy data Z, given the loss function ℓ, is defined as the expectation of ℓ:

$$\mathbb{L}[Z \mid W, \ell] \;=\; \mathbb{E}[\ell \mid W, Z] \;=\; \sum_{wz} P_{W,Z}(w, z) \ell(w, z). \tag{4.6}$$

We will omit ℓ when it is clear from the context. Note that, given a data model $P_{X,W}$, the utility loss can be expressed in terms of the mechanism $P_{Z|W}$:

$$\mathbb{L}[Z \mid W] \;=\; \sum_{xwz} P_{X,W}(x,w)P_{Z|W}(z|w)\ell(w,z). \tag{4.7}$$

Our goal is to build a privacy-protection mechanism that keeps the loss below a certain threshold L. We denote by M_L the set of such mechanisms, namely:

$$M_L \;\stackrel{\text{def}}{=}\; \{P_{Z|W} \mid \mathbb{L}[Z \mid W] \leqslant L\}. \tag{4.8}$$

The following property is immediate:

Proposition 3 (Convexity of M_L)**:** The set M_L is convex and closed.

4.2.2 Quantifying privacy as mutual information

For the notions of entropy, residual entropy, and mutual information we refer to section 2.2 for a deeper discussion. We briefly recall the basic information-theoretic definitions of cross-entropy that will be used in the next sections.
The cross-entropy between the posterior and the prediction:

$$CE(X,Y) = -\sum_z P_Z(z) \sum_x P_{X|Z}(x|z) \log P_{Y|Z}(y|z). \tag{4.9}$$

For the notion of entropy, residual entropy, and mutual information we refer to section 2.2 for a deeper discussion.

We recall that the more correlated X and Y are, the larger is $I(X;Y)$, and viceversa. The minimum $I(X;Y) = 0$ is when X and Y are independent; the maximum is when the value of X determines uniquely the value of Y and viceversa. In contrast, $CE(X,Y)$, that represents the precision loss in the classification prediction, is not related to the correlation between X and Y, but rather to the similarity between $P_{X|Z}$ and $P_{Y|Z}$: the more similar they are, the smaller is $CE(X,Y)$. In particular, the minimum $CE(X,Y)$ is when $P_{X|Z} = P_{Y|Z}$.

The privacy leakage of a mechanism $P_{Z|W}$ with respect to an attacker C, characterized by the prediction $P_{Y|Z}$, will be quantified by the mutual information $I(X;Y)$. This notion of privacy will be used as objective function, rather than the more typical cross entropy $CE(X,Y)$. As explained in the introduction, this choice makes the training of G more stable because, in order to reduce $I(X;Y)$, G cannot simply swap around the labels of the classification learned by C, it must reduce the correlation between X and Z (via suitable modifications of $P_{Z|W}$), and in doing so it limits the amount of information that *any* adversary can infer about X from Z. We will come back on this point in more detail in section 4.3.1.

4.2.3 Formulation of the game

The game that G and C play corresponds to the following minimax formulation:

$$\min_{G} \max_{C} I(X;Y) \tag{4.10}$$

where the minimization by G is on the mechanisms $P_{Z|W}$ ranging over M_L, while the maximization by C is on the classifications $P_{Y|Z}$.

Note that $P_{Z|W}$ can be seen as a stochastic matrix and therefore as an element of a vector space. An important property for our purposes is that the mutual information is convex with respect to $P_{Z|W}$:

Proposition 4 (Convexity of I)**:** Given $P_{X,W}$ and $P_{Y|Z}$, let $f(P_{Z|W}) = I(X;Y)$. Then f is convex.

Proof. Let us recall that

$$X \leftrightarrow W \leftrightarrow Z \leftrightarrow Y. \tag{4.11}$$

represents a Markov chain where:

- the relation between the two random variables X and W is defined by the data distribution $P_{X,W}$,
- the relation between Z and Y depends only on the chosen classifier according to $P_{Y|Z}$,
- the relation between Z and W can be described by the variable $P_{Z|W}$

If we consider X as the secret input and Y as the observable output of a stochastic channel, the mutual information between the two random variable can be expressed as

$$I(X;Y) = g(P_{Y|X}). \tag{4.12}$$

We know from [CT06] that $g(\cdot)$ is a convex function with respect to $P_{Y|X}$. We can express $P_{Y|X}$ as:

$$P_{Y|X}(y|x) = \frac{\sum_{zw} P_{X,W}(x,w) P_{Z|W}(z|w) P_{Y|Z}(y|z)}{\sum_{w} P_{X,W}(x,w)}. \tag{4.13}$$

Equation (4.13) represents a linear function of the variable $P_{Z|W}$ (all the other probabilities are constant). Hence $f(P_{Z|W}) = g(h(P_{Z|W}))$ where $g(\cdot)$ is convex and $h(\cdot)$ is linear. The composition of a convex function with a linear one is a convex function and this concludes the proof. □

Proposition 3 and proposition 4 show that this problem is well defined: for any choice of C, $I(X;Y)$ has a global minimum in M_L, and no strictly-local minima.

On the use of the the classifier

We note that, in principle, one could avoid using the GAN paradigm, and try to achieve the optimal mechanism by solving, instead, the following minimization problem:

$$\min_G I(X;Z) \tag{4.14}$$

where $\min_G I(X;Z)$ is meant, as before, as a minimization over the mechanisms $P_{Z|W}$ ranging over M_L. This approach would have the advantage that it is independent from the attacker, so one would need to reason only about G (and there would be no need for a GAN).

The main difference between $I(X;Y)$ and $I(X;Z)$ is that the latter represents the information about X available to any adversary, not only those that are trying to retrieve X by building a classifier. This fact reflects in the following relation between the two formulations:

Proposition 5:

$$\min_G \max_C I(X;Y) \leqslant \min_G I(X;Z)$$

Proof. Given that eq. (4.1) represents a Markov chain, $X \leftrightarrow Z \leftrightarrow Y$ represents one as well. From the data processing inequality it follows that:

$$I(X;Y) \leqslant I(X;Z)\,. \tag{4.15}$$

Hence we have:

$$\max_C I(X;Y) \leqslant I(X;Z)\,, \tag{4.16}$$

and therefore:

$$\min_G \max_C I(X;Y) \leqslant \min_G I(X;Z)\,. \tag{4.17}$$

□

Note that, since $\min_G I(X;Z)$ is an upper bound of our target, it imposes a limit on $\max_C I(X;Y)$.

On the other hand, there are some advantages in considering $\min_G \max_C I(X;Y)$ instead than $\min_G I(X;Z)$: first of all, Z may have a much larger and more complicated domain than Y, so performing the gradient descent on $I(X;Z)$ could be infeasible. Second, if we are interested in considering only classification-based attacks, then $\min_G \max_C I(X;Y)$ should give a better result than $\min_G I(X;Z)$. In this work we focus on the former, and leave the exploration of an approach based on $\min_G I(X;Z)$ as future work.

4.2.4 Measuring privacy as Bayes error

As explained in the introduction, we intend to evaluate the resulting mechanism also in terms of Bayes error. Here we give the relevant definitions and properties.

Definition 3 (Bayes error)**:** The Bayes error of X given Y is:

$$B(X \mid Y) \;=\; \sum_y P_Y(y)(1 - \max_x P_{X|Y}(x \mid y)).$$

Namely, the Bayes error is the expected probability of "guessing the wrong id" of an adversary that, when he sees that C produces the id y, it guesses the id x that has the highest posterior probability given y.

The definition of $B(X \mid Z)$ is analogous. Given a mechanism $P_{Z|W}$, we regard $B(X \mid Y)$ as a measure of the privacy of $P_{Z|W}$ with respect to one-try [Smi09] classification-based attacks, whereas $B(X \mid Z)$ is with respect to any one-try attack. The following proposition shows the relation between the two notions.

Proposition 6: $B(X \mid Z) \leqslant B(X \mid Y)$

Proof.

$$\begin{aligned}
B(X \mid Z) &= \sum_z P_Z(z)(1 - \max_x P_{X|Z}(x \mid z)) \\
&= 1 - \sum_z P_Z(z) \max_x P_{X|Z}(x \mid z) \\
&= 1 - \sum_y P_Y(y) \sum_z P_{Z|Y}(z|y) \max_x P_{X|Z}(x \mid z) \\
&\leqslant 1 - \sum_y P_Y(y) \max_x \sum_z P_{Z|Y}(z|y) P_{X|Z}(x \mid z) \\
&= 1 - \sum_y P_Y(y) \max_x P_{X|Y}(x \mid y) \\
&= B(X \mid Y)
\end{aligned}$$

□

4.3 Proposed method

In this section we describe the implementation of our adversarial game between G and C in terms of alternate training of neural networks.

The scheme of our game is illustrated in fig. 4.1, where:

- x, y, z and w are instances of the random variables X,Y, Z and W respectively, whose meaning is described in previous section. We assume that the domains of X and Y coincide.
- s (seed) is a randomly-generated number in $[0, 1)$.
- g is the function learnt by G, and it represents an obfuscation mechanism $P_{Z|W}$. The input s provides the randomness needed to generate random noise. It is necessary because a neural network in itself is deterministic.

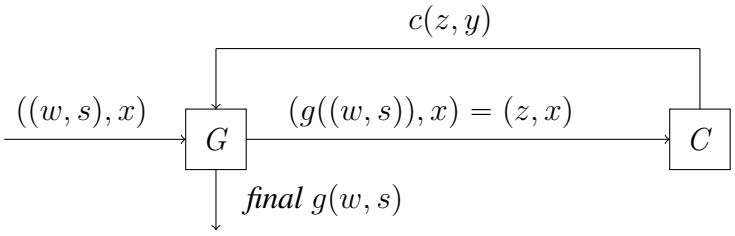


Figure 4.1 – Scheme of the adversarial nets for our setting.

- c is the classification learnt by C, corresponding to $P_{Y|Z}$.

Our paradigm has been inspired by the *GAN* [GPAM$^+$14], but it comes with some fundamental differences:

- C is a classifier performing re-identification while in the *GAN* there is a discriminator able to distinguish a real data distribution from a generated one;
- in the *GAN* paradigm the generator network tries to reproduce the original data distribution to fool the discriminator. A huge difference is that, in our adversarial scenario, G does not have a model distribution to refer to. The final data distribution only depends on the evolution of the two networks over time and it is driven by the constraints imposed in the loss functions that rule the learning process.
- We still adopt a training algorithm which alternates the training of G and of C, but as we will show in Section section 4.3, it is different from the one adopted for *GAN*.

The evolution of the adversarial network is described in algorithm 2. C and G are trained at two different moments within the same adversarial training iteration. In particular C_i is obtained by training the network C against the noise generated by G_{i-1} and G_i is obtained by fighting against C_i.

Note that in our method each C_i is trained on the output of G_{i-1}. This is a main difference with respect to the GAN paradigm, where the discriminator is trained both on the output of the generator and on samples from the target distribution generated by an external source. Another particularity of our method is that at the end of the i-th iteration, while G_i is retained for the next iteration, C_i is discarded and the classifier for iteration $i + 1$ is reinitialized to the base one C_0. The reason is that restarting from C_0 is more efficient than starting from the last trained classifier C_i. This is because G_i may have changed at step i the noise mechanism $P_{Z|W}$ and therefore the association between X and Z expressed by $P_{X|Z}$. The predictions $P_{Y|Z}(x \mid z)$ that C_i had produced during its training (trying to match the $P_{X|Z}(x \mid z)$ previously produced by G_{i-1} as closely as possible), not only is not optimal anymore: for some z's it may have become completely wrong, and starting from a wrong prediction is a drawback that slows down the learning of the new prediction. There may be several z's for which the old prediction is a good approximation of the new one to be

Algorithm 2: Adversarial algorithm with classifier reset.

Data: $train\_data$ `// Training data`
Models: G_i generator evolution at the i–th step;
C_i classifier evolution at the i–th step.

$train(n, d)$ trains the network n on the data d.
$G_i(data)$ outputs a noisy version of $data$.

C_0 = base classifier model
G_0 = base generator model
i = 0
while *True* **do**
  i += 1
  `// Train class. from scratch`
  $C_i = train(C_0, G_{i-1}(train\_data))$
  $A = G_{i-1}$ and C_i in cascade
  $A = train(A, train\_data)$
  G_i = generator layer in A

learned, but according to our experiments the net effect is negative: the training of the new classifier is usually faster if we restart from scratch. It is worth noting that this is only a matter of efficiency though: eventually, even if we started from C_i, the new classifier would "unlearn" the old, wrong predictions and learn the correct new ones.

At the end of each training iteration we evaluate the quality of the produced noise by checking the performance of the C network. In particular we make sure that the noise produced by the G network affects the training, validation and test data in a similar way. In fact, in case the performances were good on the training data but not on the the other data, this would be a result of over-fitting rather than of a quality indicator of the injected noise.

We describe now in more detail some key implementation choices of our proposal.

4.3.1 Mutual information vs cross entropy

Based on the formulation of our game eq. (4.10), the alternate training of both G and C is performed using the *mutual information* $I(X;Y)$ as the loss function. The goal of G is to minimize $I(X;Y)$ by refining the mechanism $P_{Z|W}$, while C aims at maximizing it by refining the classifier $P_{Y|Z}$.

We remark that the use of mutual information as loss function is not standard. A more typical function for training a classifier is the cross entropy $CE(X,Y)$, which is more efficient to implement. $CE(X,Y)$ is minimized when $P_{X|Z}$ and $P_{Y|Z}$ coincide. Such outcome would correspond to the perfect classifier, that predicts the exact probability $P_{X|Z}(x|z)$ that a given sample z belongs to the class x. One could then think of reformulating the game in terms of the cross entropy $CE(X,Y)$, where C would be the minimizer (trying to infer prob-

abilistic information about the secret x from a given observation z) and G the maximizer (trying to prevent the adversary C from achieving this knowledge). However, as already observed in example 1 in the introduction, training G via $CE(X,Y)$ does not allow to reach an equilibrium, because it takes into account only one adversarial strategy (i.e., one particular classification). Indeed, a maximum $CE(X,Y)$ can be achieved with a $P_{Z|W}$ that simply causes a swapping of the associations between the labels x's and the corresponding noisy locations z's. This would change $P_{X|Z}$ and therefore fool the present classifier (because the prediction $P_{Y|Z}$ would not be equal anymore to $P_{X|Z}$), but at the next round, when C will be trained on the new data, it will learn the new classification $P_{X|Z}$ and obtain, again, the maximum information about x that can be inferred from z. The possibility of ending up in such cyclic behavior is experimentally proved in section 4.4.1. Note that this problem does not happen with mutual information, because swapping the labels does not affect $I(X;Y)$ at all.

Since G can only change the mechanism $P_{Z|W}$, the only way for G to reduce the mutual information $I(X;Y)$ is to reduce $I(X;Z)$ by reducing the correlation between W and Z (X is correlated to Z only via W) . This limits the information about X that can be inferred from Z, for any possible adversary, i.e., for any possible prediction $P_{Y|Z}$, hence also for the optimal one. Still, if Z is very large $I(X;Z)$ cannot be reduced directly in an efficient way, and this is the reason why G needs the feedback of the optimal prediction $P_{Y|Z}$: in contrast to $I(X;Z)$, minimizing $I(X;Y)$ can be done effectively in neural networks via the gradient descent when $\mathcal{X}$ (the domain of X and Y) is "reasonably small".

The above discussion about $I(X;Y)$ vs $CE(X,Y)$ holds for the generator G, but what about the adversary C? Namely, for a given $P_{Z|W}$, is it still necessary to train C on $I(X;Y)$, or could we equivalently train it on $CE(X,Y)$? The following result answers this question positively.

Proposition 7:

$$\operatorname*{argmin}_G \max_C \; I(X;Y) = \operatorname*{argmin}_G I(X,Y') \; ,$$

with Y' defined by $P_{Y'|Z} = \operatorname*{argmin}_C \; CE(X,Y') = P_{X|Z}$.

Proof. P_{XW} is fixed, and therefore $H(X)$ is fixed as well. Hence the goal of C of maximizing $I(X;Y)$ reduces to maximizing $-H(X|Y)$. Consider two mechanisms, $P_{Z_1|W}$ and $P_{Z_2|W}$, and the distributions induced on X by Z_1 and Z_2 respectively, namely $P_{X|Z_1}$ and $P_{X|Z_2}$. Consider the predictions $P_{Y_1|Z_1}$ and $P_{Y_1|Z_2}$ that C obtains by minimizing the cross entropy with $P_{X|Z_1}$ and $P_{X|Z_2}$ respectively.

It is well known that $\operatorname{argmin}_Q CE(P,Q) = P$, hence we have $P_{Y_1|Z_1} = P_{X|Z_1}$ and $P_{Y_2|Z_2} = P_{X|Z_2}$. (Note that X, Y_1 and Y_2 all have the same domain $\mathcal{X}$.) Hence, taking into

account that $X \leftrightarrow Z_1 \leftrightarrow Y_1$ and $X \leftrightarrow Z_2 \leftrightarrow Y_2$ (i.e., they are Markov chains), we have:

$$-H(X|Y_1) \leqslant -H(X|Y_2)$$

iff

$$\begin{array}{c} \sum_z P_{Z_1}(z) \sum_{xy} P_{X|Z_1=z}(x|z) P_{Y_1|Z_1=z}(y|z) \log P_{Y_1|Z_1=z}(y|z) \\ \leqslant \\ \sum_z P_{Z_2}(z) \sum_{xy} P_{X|Z_2=z}(x|z) P_{Y_2|Z_2=z}(y|z) \log P_{Y_2|Z_2=z}(y|z) \end{array}$$

iff

$$\begin{array}{c} \sum_z P_{Z_1}(z) \sum_{xy} P_{X|Z_1=z}(x|z) P_{X|Z_1=z}(y|z) \log P_{X|Z_1=z}(x|z) \\ \leqslant \\ \sum_z P_{Z_2}(z) \sum_{xy} P_{X|Z_2=z}(x|z) P_{X|Z_2=z}(y|z) \log P_{X|Z_2=z}(x|z) \end{array}$$

iff

$$-H(X|Z_1) \leqslant -H(X|Z_2).$$

Finally, observe that

$$-H(X|Z_1) \leqslant -H(X|Z_2) \quad \text{implies} \quad \max_{P_{Y_1|Z}} I(X;Y_1) \leqslant \max_{P_{Y_2|Z}} I(X;Y_2) \tag{4.18}$$

and recall that $P_{Y_i|Z}$ is the prediction produced by C. □

Given the above result, and since minimizing $CE(X,Y)$ is more efficient than maximizing $I(X;Y)$, in our implementation we have used $CE(X,Y)$ for the training of C. Of course, we cannot do the same for G: as discussed above, the generator needs to be trained by using $I(X;Y)$.

A consequence of proposition 7 is that the adversary represented by C at the point of equilibrium is at least as strong as the Bayesian adversary, namely the adversary that minimizes the expected probability of error in the 1-try attack (which consists in guessing a single secret x given a single observable z [Smi09].) Indeed, from $P_{Y|Z}$ one can derive the following decision function (deterministic classifier) $f^* : \mathcal{Z} \to \mathcal{X}$, which assigns to any z the class y with highest predicted probability:

$$f^*(z) \;=\; \operatorname*{argmax}_y P_{Y|Z}(y|z) \tag{4.19}$$

To state formally the property of the optimality of f^* with respect to 1-try attacks, let us recall the definition of the expected error $R(f)$ for a generic decision function $f : \mathcal{Z} \to \mathcal{X}$:

$$R(f) \;=\; \sum_{xz} P_{X,Z}(x,z) \overline{\mathbb{1}}_f(x,z) \tag{4.20}$$

where

$$\overline{\mathbb{1}}_f(x,z) \;=\; \begin{cases} 1 & \text{if } f(z) \neq x \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

We can now state the following result, that relates the error of the attacker f^* (induced by the C at the equilibrium point) and the minimum Bayes error of any adversary for the G at the equilibrium point (cfr. definition 3 and proposition 6):

Proposition 8: If $P_{Y|Z} = \operatorname{argmin}_C \; CE(X,Y)$, and f^* is defined as in eq. (4.19), then:

$$R(f^*) = B(X,Z)$$

Proof. Let $P_{Y|Z} = \operatorname{argmin}_C \; CE(X,Y)$ and let f^* be defined as in eq. (4.19). We note that, for every $z \in \mathcal{Z}$:

$$\begin{aligned}
\sum_x P_{X|Z}(x|z)\overline{\mathbb{1}}_{f^*}(x,z) &= \sum_{x \neq f^*(z)} P_{X|Z}(x,z) \\
&= \sum_{x \neq \operatorname{argmax}_y P_{Y|Z}(y|z)} P_{X|Z}(x,z) \\
&= 1 - \sum_{x = \operatorname{argmax}_y P_{Y|Z}(y|z)} P_{X|Z}(x,z) \\
&= 1 - P_{X|Z}(\operatorname*{argmax}_x P_{X|Z}(x|z) \mid z)) \\
&= 1 - \max_x P_{X|Z}(x \mid z))
\end{aligned}$$

where the first equality is due to the definition of $\overline{\mathbb{1}}_{f^*}$, the second one is due to the definition of f^*, and the last but one follows from the fact that $P_{Y|Z} = \operatorname{argmin}_C \; CE(X,Y)$ and therefore, by proposition 7, $P_{Y|Z} = P_{X|Z}$. Hence, we have:

$$\begin{aligned}
R(f^*) &= \sum_{xz} P_{X,Z}(x,z)\overline{\mathbb{1}}^*_f(x,z) \\
&= \sum_{xz} P_Z(z)P_{X|Z}(x|z)\overline{\mathbb{1}}^*_f(x,z) \\
&= \sum_z P_Z(z) \sum_x P_{X|Z}(x|z)\overline{\mathbb{1}}^*_f(x,z) \\
&= \sum_z P_Z(z)(1 - \max_x P_{X|Z}(x \mid z)) \\
&= B(X \mid Z) \qquad \text{(cfr. definition 3)}
\end{aligned}$$

□

4.3.2 Mutual information: implementation

In order to describe the implementation of the mutual information loss function, we will consider the training on a specific batch of data. This technique is based on the idea that the whole training set of cardinality N can be split into subsets of cardinality N' with $N' \leqslant N$. This is useful to make the data fit into the memory and, since during each epoch the network is trained on all the batches, this corresponds to using all the training data (provided that the data distribution in each batch is a high fidelity representation of the training set distribution, otherwise the learning could be unstable).

To obtain the mutual information between X and Y we estimate the distributions P_X, P_Y and $P_{X,Y}$. Then we can compute $I(X;Y)$ using eq. (2.8), or equivalently as the formula:

$$\sum_x P_X(x) \log P_X(x) - \sum_{x,y} P_{X,Y}(x,y) \log \frac{P_{X,Y}(x,y)}{P_Y(y)}. \tag{4.22}$$

Let us consider a batch consisting of N' samples of type (z,x) in the context of the classification problem, and let $|\mathcal{X}|$ represents the cardinality of $\mathcal{X}$, i.e., the total number of classes. In the following we denote by T and Q, respectively, the target and the prediction matrices for the batch. Namely, T and Q are $N' \times |\mathcal{X}|$ matrices, whose rows correspond to samples and whose columns to classes, defined as follows. T represents the class one-hot encoding: the element in row i and column x, $T(i,x)$, is 1 if x is the target class for the sample i, and 0 otherwise. Q, on the other hand, reports the probability distribution over the classes computed by the classifier: $Q(i,x)$ is the predicted probability that sample i be in class x.

The estimation of $P_X(x)$ for the given batch can be obtained by computing the frequency of x among the samples, namely:

$$P_X(x) = \frac{1}{N'} \sum_{i=1}^{N'} T(i,x). \tag{4.23}$$

Similarly, $P_Y(y)$ is estimated as the expected prediction of y:

$$P_Y(y) = \frac{1}{N'} \sum_{i=1}^{N'} Q(i,y). \tag{4.24}$$

The joint distribution $P_{X,Y}$ can be estimated by considering the correlation of X and Y through the samples. Indeed, the probability that sample i has target class x and predicted class y can be computed as the product $T(i,x)Q(i,y)$, and by summing up the contributions of all samples (where each sample contributes for $1/N'$) we obtain $P_{X,Y}(x,y)$.

More precisely, for a sample $i \in \{1,...,N'\}$ let us define the $|\mathcal{X}| \times |\mathcal{X}|$ matrix J_i as $J_i(x,y) = T(i,x)Q(i,y)$. Then we can estimate $P_{X,Y}(x,y)$ as:

$$P_{X,Y}(x,y) = \frac{1}{N'} \sum_{i=1}^{N'} J_i(x,y). \tag{4.25}$$

The estimation of the mutual information relies on the estimation of the probabilities, which is based on the computation of the frequencies. Hence, in order to obtain a good estimation, the batches should be large enough to represent well the true distributions. Furthermore, if the batch size is too small, the gradient descent is unstable since the representation of the distribution changes from one batch to the other. In the ML literature there are standard validation techniques (such as the *cross validation*) that provide guidelines to achieve a "good enough" estimation of the probabilities.

Base models

The base model C_0 is simply the "blank" classifier that has not learnt anything yet (i.e. the weights are initialized according to the Glorot initialization, which is a standard initialization technique [GB10]). As for G_0, we have found out experimentally that it is convenient to start with a noise function pretty much spread out. This is because in this way the generator has more data points with non-null probability to consider, and can figure out faster which way to go to minimize the mutual information.

4.3.3 Utility

The utility constraint is incorporated in the loss function of G in the following way:

$$Loss_G = \alpha \times Loss_{utility} + \beta \times I(X;Y), \tag{4.26}$$

where α and β are parameters that allow us to tune the trade-off between utility and privacy. The purpose of $Loss_{utility}$ is to ensure that the constraint on utility is respected, i.e., that the obfuscation mechanism that G is trying to produce stays within the domain M_L. We recall that M_L represents the constraint $\mathbb{L}[Z \mid W] \leqslant L$ (cfr. eq. (4.8)). Since we need to compute the gradient on the loss, we need a derivable function for $Loss_{utility}$. We propose to implement it using $\mathrm{softplus}$, which is a function of two arguments in $\mathbb{R}$ defined as: $\mathrm{softplus}(a,b) = \ln(1 + e^{(a-b)})$. This function is non negative, monotonically increasing, and its value is close to 0 for $a < b$, while it grows very quickly for $a > b$. Hence, we define

$$Loss_{utility}(P_{Z|W}) = \mathrm{softplus}(\mathbb{L}[Z \mid W], L). \tag{4.27}$$

With this definition, $Loss_{utility}$ does not interfere with $I(X;Y)$ when the constraint $\mathbb{L}[Z \mid W] \leqslant L$ is respected, and it forces G to stay within the constraint because its growth when the constraints is not respected is very steep.

4.3.4 On the convergence of our method

In principle, at a each iteration i, our method relies on the ability of the network G_i to improve the obfuscation mechanism starting from the one produced by G_{i-1}, and given only the original locations and the model C_i, which are used to determine the direction of the

gradient for $Loss_G$. The classifier C_i is a particular adversary modeled by its weights and its biases. However, thanks to the fact that the main component of $Loss_G$ is $I(X;Y)$ and not the the cross entropy, G_i takes into account all the attacks that would be possible from C_i's information. We have experimentally verified that indeed, using the mutual information rather than the cross entropy, determines a substantial improvement on the convergence process, and the resulting mechanisms provide a better privacy (for the same utility level). Again, the reason is that the the cross entropy would be subject to the "swapping effect" illustrated by example 1 in the introduction.

Another improvement on the convergence is due the fact that, as explained before, we reset the classifier to the initial weight setting (C_0) at each iteration, instead than letting C_i evolve from C_{i-1}.

The function that G has to minimize, $Loss_G$, is convex with respect to $P_{Z|W}$. This means that there are only global minima, although there can be many of them, all equivalent. Hence for sufficiently small updates the noise distribution modeled by $P_{Z|W}$ converges to one of these optima, provided that the involved network has enough capacity to compute the gradient descent involved in the training algorithm. In practice, however, the network G represents a limited family of noise distributions, and instead of optimizing the noise distribution itself we optimize the weights of this network, which introduces multiple critical points in the parameter space.

Number of epochs and batch size

The convergence of the game can be quite sensitive to the number of epochs and batch size. We just give two hints here, referring to literature [KMN+16] for a general discussion about the impact they have on learning.

First, choosing a batch too small for training G might result in too strict a constraint on the utility. In fact, since the utility loss is an expectation, a larger number of samples makes it more likely that some points are pushed further than the threshold, taking advantage of the fact that their loss may be compensated by other data points for which the loss is small.

Second, training C for too few epochs might result into a too weak adversary. On the other hand if it is trained for a long time we should make sure that the classification performances do not drop over the validation and test set because that might indicate an over-fitting problem.

4.4 Cross Entropy vs Mutual Information: experiments on synthetic data

In this section we perform experiments on a synthetic dataset to obtain an intuition about the behaviour of our method. The dataset is constructed with the explicit purpose of being simple, to facilitate the interpretation of the results. The main outcome of these experiments is confirming the fact that, as discussed in section 4.3.1, training the generator G wrt *cross entropy is not sound*. Even in our simple synthetic case, training G with $CE(X,Y)$ as

the loss function fails to converge: G is just "moving points around", temporarily fooling the *current* classifier, but failing to really hide the correlation between the secrets and the reported locations.

On the other hand, training G with mutual information behaves as expected: the resulting network generates noise that mixes all classes together, making the classification problem hard for *any* adversary, not only for the *current* one. Note that cross entropy is still used, but only for C (cfr. section 4.3.1).

The dataset

We consider a simple location privacy problem; 4 users $\mathcal{X} = \mathcal{Y} = \{blue, red, green, yellow\}$ want to disclose their location while protecting their identities. Both the real locations $\mathcal{W}$ as well as the reported locations $\mathcal{Z}$ are taken to be all locations in a squared region of 6.5×6.5 sq km centered in 5, Boulevard de Sébastopol, Paris. Each location entry is defined by a pair of coordinates normalized in $[-1, 1]$.

The synthetic dataset consists of 600 real locations for each of the 4 users (classes), for a total of 2400 entries. The locations of each user are placed around one of the vertices of a square of 300×300 sq meters centered in 5, Boulevard de Sébastopol, Paris. (Each user corresponds to a different vertex.) They are randomly generated so to form a cloud of 600 entries around each vertex and in such a way that no locations falls further than about 45m from the corresponding vertex. These sets are represented in fig. 4.2 ((a) and (b), left): it is evident from the figure that the four classes are easily distinguishable; without noise a linear classifier could predict the class of each location with no error at all.

Of the total 2400 entries of the dataset we use 1920 for training and validation (480 for each user) and 480 for testing (120 for each user).

Network architecture

A relatively simple architecture is used for both G and C networks. They consist of three fully connected hidden layers of neuron with ReLU function. In particular C has 60, 100 and 51 hidden neurons respectively in the first, second and third hidden layers. The G network has 100 neurons in each hidden layer; such an architecture has proved to be enough to learn how to reproduce the Laplace noise distribution ($\epsilon = \ln(2)/100$) with a negligible loss.

Bayes error estimation

As explained in section 4.2, we use the Bayes error $B(X \mid Z)$ to evaluate the level of protection offered by a mechanism. To this purpose, we discretize $\mathcal{Z}$ into a grid over the 6.5×6.5 sq km region, thus determining a partition of the region into a number of disjoint *cells*. We will create different grid settings to see how the partition affects the Bayes error. In particular, we will consider the cases where the side of a cell is 25m, 50m, 100m and

500m long, which corresponds to $260 \times 260 = 67600$, $130 \times 130 = 16900$, $65 \times 65 = 4225$ and $13 \times 13 = 169$ cells, respectively.

We run experiments with different numbers of obfuscated locations (hits). Specifically, for each grid we consider 10, 100, 200 and 500 obfuscated hits for each original one.

Each hit falls in exactly one cell. Hence, we can estimate the probability that a hit is in cell i as:

$$P(cell_i) \;=\; \frac{\text{number of hits in } cell_i}{\text{total number of hits}}, \tag{4.28}$$

and the probability that a hit in cell i belong to class j:

$$P(Class_j|cell_i) \;=\; \frac{\text{number of hits of } class_j \text{ in } cell_i}{\text{number of hits in } cell_i}, \tag{4.29}$$

We can now estimate of the Bayes error as follows:

$$B(X \mid Z) = 1 - \sum_{i=0}^{k-1} \max_j \; P(Class_j|cell_i)P(cell_i) \tag{4.30}$$

where k is the total number of cells.

Note that these computations are influenced by the chosen grid. In particular we have two extreme cases:

- when the grid consists of only one cell the Bayes error is $1 - 1/k = {k-1}/k$ for any obfuscation mechanism $P_{Z|W}$.
- when the number of cells is large enough so that each cell contains at most one hit, then the Bayes error is 0 for any obfuscation mechanism.

In general, we expects a finer granularity to give higher discrimination power and to decrease the Bayes error, especially with methods that scatter the obfuscated locations far away.

We estimate the Bayes error on the testing data in order to evaluate how well the obfuscation mechanisms protect new data samples never seen during the training phase. Moreover we evaluate the Bayes error on the same data we used for training and we compare the results with those obtained for the testing data. We notice that, in general, the difference between the two results is not large, meaning that the deployed mechanisms efficiently protect the new samples as well.

The planar Laplace mechanism

We compare our method against the planar Laplace mechanism [ABCP13], whose probability density to report z, when the true location is w, is:

$$\mathcal{L}^{\epsilon}_{w}(z) = \frac{\epsilon^2}{2\pi}\, e^{-\epsilon\, d(w,z)}\,, \tag{4.31}$$

where $d(w, z)$ is the Euclidean distance between w and z.

In order to compare the the Laplace mechanism with ours, we need to tune the privacy parameter ϵ so that the expected distortion of $\mathcal{L}^\epsilon$ is the same as the upper bound on the utility loss applied in our method, i.e. L. To this purpose, we recall that the expected distortion $\mathbb{L}[Z \mid W]$ of the planar Laplace depends only on ϵ (not on the prior P_W), and it is given by:

$$\mathbb{L}[Z \mid W] \;=\; \frac{2}{\epsilon}. \tag{4.32}$$

4.4.1 Experiment 1: relaxed utility constraint

As a first experiment, we choose for the upper bound L on the expected distortion a value high enough so that in principle we can achieve the highest possible privacy, which is obtained when the observed obfuscated location gives no information about the true location, which means that $I(X;Y) = 0$. In this case, the attacker can only do random guessing. Since we have 4 users, the Bayes error is $B(X \mid Y) = 1 - 1/4 = 0.75$.

For the distortion, we take $\ell(w, z)$ to be the geographical distance between w and z. One way to achieve the maximum privacy is to map all locations into the middle point. To compute a sufficient L, note that the vertices of the original locations form a square of side 300m, hence each vertex is at a distance $300 \times \sqrt{2}/2 \approx 212$m from the center. Taking into account that the locations can be as much as 45m away from the corresponding vertex, we conclude that any value of L larger than $212 + 45 = 247$m should be enough to allow us to obtain the maximum privacy. We set the upper bound on the distortion a little higher:

$$L \;=\; 270\text{m}, \tag{4.33}$$

but we will see from the experiments that a much smaller value of L would have been sufficient.

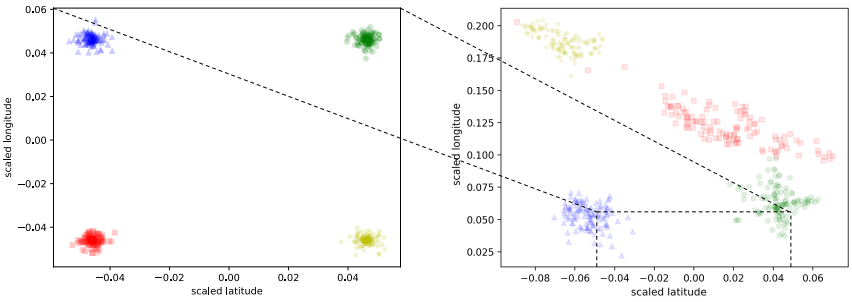
We now need to tune the planar Laplace so that the expected distortion is at least L. We decide to set:

$$\epsilon = \frac{\ln 2}{100} \tag{4.34}$$

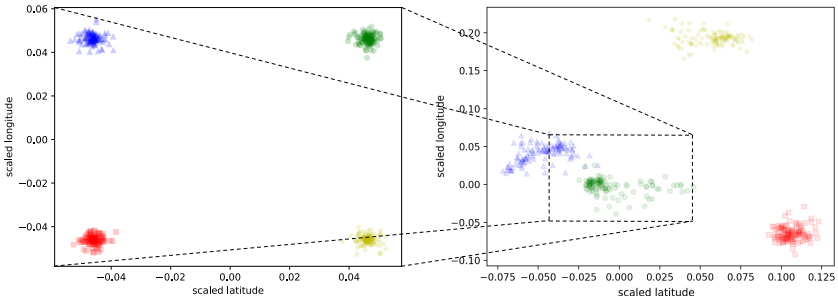
which, using eq. (4.32), gives us a value

$$\mathbb{L}[Z \mid W] \approx 288\text{m} > L. \tag{4.35}$$

We have used this instance of the planar Laplace also as a starting point of our method: we have defined G_0 as $\mathcal{L}^\epsilon$ with $\epsilon = \ln 2/100$. For the next steps, G_i and C_i are constructed as explained in Algorithm algorithm 2. In particular, we train the generator with a batch size of 128 samples for 100 epochs during each iteration. The learning rate is set to 0.0001. For this particular experiment we set the weight for the utility loss to 1 and the weight for the mutual information to 2. The classifier is trained with a batch size of 512 samples and 3000 epochs for each iteration. The learning rate for the classifier is set to 0.001.



(a) Iteration 30



(b) Iteration 40

Figure 4.2 – Using cross entropy to produce the noise does not make the system converge. The left sides of fig. 4.2a and fig. 4.2b show the original synthetic data without noise. The right sides show the noisy data at different iterations. $L = 270$m.

Training G with the cross entropy

As discussed in Sec section 4.3.1, training G wrt $CE(X, Y)$ is not sound. This is confirmed in the experiments by the fact that G is failing to converge. Figure 4.2 shows the distribution generated by G in two different iterations of the game. We observe that, trying to fool the classifier C, the generator on the right-hand side has simply moved locations around, so that each class has been placed in a different area. This clearly confuses a classifier trained on the distribution of the left-hand side, however the correlation between labels and location is still evident. A classifier trained on the new G can infer the labels as accurately as before.

As a consequence, after each iteration, the accuracy of the newly trained C_i is always 1, while the Bayes error $B(X|Z)$ is 0. The generator fails to converge to a distribution that effectively protects the users' privacy. We can hence conclude that the use of cross entropy is unsound for training G.

Training G wrt mutual information

Using now $I(X;Y)$ for training G (while still using the more efficient cross entropy for C, as explained in Sec section 4.3.1), we observe a totally different behaviour. After each iteration the accuracy of the classifier drops, showing that the generator produces meaningful noise.

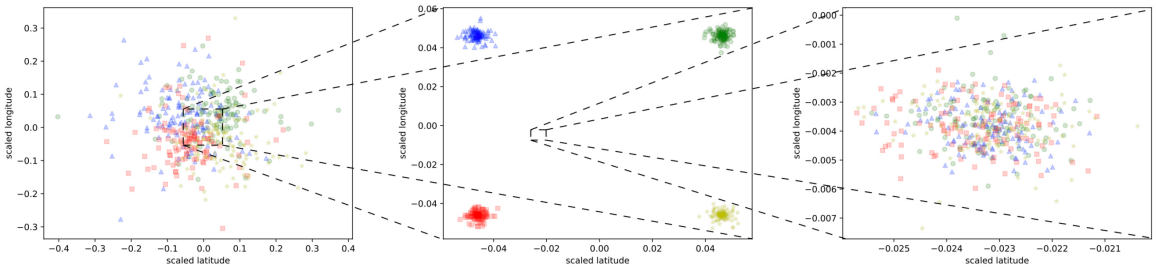


Figure 4.3 – Synthetic testing data. From left to right: Laplace noise, no noise, our noise produced using mutual information. $L = 270$m.

Number of cells			
13×13	65×65	130×130	260×260
0.75	0.00	0.00	0.00

(a) Training data.

Number of cells			
13×13	65×65	130×130	260×260
0.75	0.00	0.00	0.00

(b) Testing data.

Table 4.4 – Estimation of $B(X \mid Z)$ on the original version of the synthetic data.

Around iteration $i = 149$ the accuracy of C_i becomes ≈ 0.25 both over the training and the validation set. This means that C_i just randomly predicts one of the four classes. We conclude that the noise injection is maximally effective, since 0.75 is the maximum possible Bayes error. Hence we know that we can stop.

The result of our method, i.e., the final generator G_i, to the testing set is reported in fig. 4.3 (rightmost plot). The empirical distortion is ≈ 219.26m. This is way below the limit of 270m set in eq. (4.33), and it is due to the fact that to achieve the optimum privacy we probably do not need more than ≈ 220m. In fact, the distance of the vertices from the center is ≈ 212m, and even though some locations are further away (up to 45m more), there are also locations that are closer, and that compensate the utility loss (which is a linear average measure).

For comparison, the result of the application of the planar Laplace to the testing set is illustrated in fig. 4.3 (leftmost plot). The empirical distortion (i.e., the distortion computed on the sampled obfuscated locations) is ≈ 298.40m, which is in line with the theoretical distortion formulated in eq. (4.35).

From fig. 4.3 we can see that, while the Laplace tends to "spread out" the obfuscated locations, our method tends to concentrate them into a single point (mode collapse), i.e., the mechanism is almost deterministic. This is due to the fact that the utility constraint is

	Number of cells							
	13 × 13		65 × 65		130 × 130		260 × 260	
Obf	Lap	Our	Lap	Our	Lap	Our	Lap	Our
10	0.60	0.75	0.40	0.75	0.38	0.75	0.35	0.73
100	0.60	0.75	0.41	0.75	0.40	0.75	0.39	0.74
200	0.60	0.75	0.41	0.75	0.40	0.75	0.39	0.74
500	0.60	0.75	0.41	0.75	0.40	0.75	0.40	0.74

(a) Training data.

	Number of cells							
	13 × 13		65 × 65		130 × 130		260 × 260	
Obf	Lap	Our	Lap	Our	Lap	Our	Lap	Our
10	0.59	0.75	0.38	0.75	0.36	0.75	0.26	0.73
100	0.60	0.75	0.40	0.75	0.39	0.75	0.37	0.74
200	0.60	0.75	0.41	0.75	0.39	0.75	0.38	0.74
500	0.60	0.75	0.41	0.75	0.40	0.75	0.39	0.74

(b) Testing data.

Table 4.5 – Estimation of $B(X \mid Z)$ on synthetic data for the Laplace and our mechanisms, with $L = 270$m. The empirical utility loss for training and testing data is $\approx 282.07\text{m} - 298.40\text{m}$ respectively for the Laplace and $\approx 219.70\text{m} - 219.26\text{m}$ for ours. The optimal mechanism gives $B(X \mid Z) = 1 - 1/4 = 0.75$.

sufficiently loose to allow the noisy locations to be displaced enough so to overlap all in the same point. When the utility constraint is stricter, the mechanism is forced to be probabilistic (and the mode collapse does not happen anymore). For example, consider two individuals, A and B, in locations a and b respectively, at distance 100m, assume that $L = 40$m. Assume also, for simplicity, that there are no other locations available. Then the optimal solution maps a into b with probability $2/5$, and into itself with probability $3/5$ and vice versa for b). Nevertheless, we can expect that our mechanism will tend to overlap the obfuscated locations of different classes, as much as allowed by the utility constraint. With the Laplace, on the contrary, the areas of the various classes remain pretty separated. This is reflected by the Bayes error estimation reported in table 4.5.

We note that the Bayes error of the planar Laplace tend to decrease as the grid becomes finer. We believe that this is due to the fact that, with a coarse grid, there is an effect of confusion simply due to the large size of each cell. We remark that the behavior of our noise, on the contrary, is quite stable. Note that, when the grid is very coarse (13×13 cells) the Bayes error is 0.75 already on the original data (cfr. table 4.4), which must be due to the fact that all the vertices are in the same cell. While the Bayes error remains 0.75 also with our obfuscation mechanism, with Laplace it decreases to 0.60. The reason is that the noise scatters the locations in different cells, and they become, therefore distinguishable.

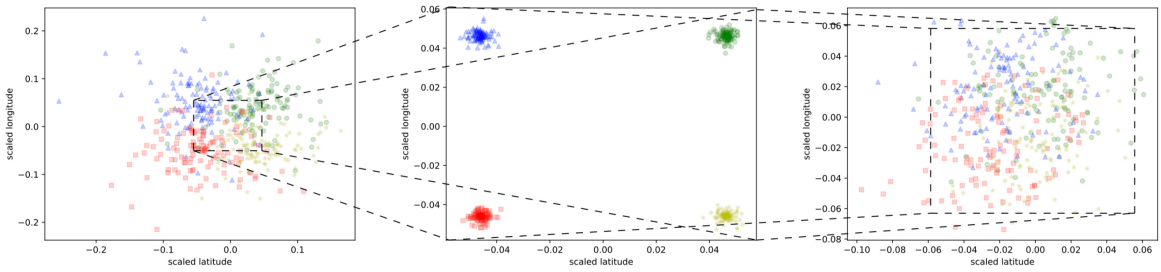


Figure 4.4 – Synthetic testing data. From left to right: Laplace noise, no noise, our noise produced using mutual information. $L = 173$m.

	Number of cells							
	13×13		65×65		130×130		260×260	
Obf	Lap	Our	Lap	Our	Lap	Our	Lap	Our
10	0.64	0.74	0.26	0.45	0.23	0.43	0.22	0.41
100	0.64	0.74	0.26	0.45	0.24	0.43	0.23	0.42
200	0.64	0.74	0.26	0.45	0.24	0.43	0.24	0.42
500	0.64	0.74	0.26	0.45	0.24	0.43	0.24	0.42

(a) Training data.

	Number of cells							
	13×13		65×65		130×130		260×260	
Obf	Lap	Our	Lap	Our	Lap	Our	Lap	Our
10	0.63	0.74	0.25	0.44	0.23	0.42	0.19	0.39
100	0.64	0.74	0.26	0.45	0.24	0.43	0.23	0.42
200	0.64	0.74	0.26	0.45	0.23	0.43	0.23	0.42
500	0.64	0.74	0.26	0.45	0.23	0.43	0.23	0.42

(b) Testing data

Table 4.6 – Estimation of $B(X \mid Z)$ on the synthetic data for the Laplace and for our mechanisms, with $L = 173$m. The empirical utility loss for training and testing data is ≈ 170.53m – 172.35m respectively for the Laplace and ≈ 166.78m – 171.50m for ours. The optimal mechanism gives $B(X \mid Z) = 0.50$, since the utility bound is large enough to let mixing the red and blue points, as well as the green and the yellow, but does not allow more confusion than that.

4.4.2 Experiment 2: stricter utility constraint

We are now interested in investigating how our method behaves when a stricter constraint on the utility loss is imposed. In order to do so, we run an experiment similar to the one in section 4.4.1. We repeat the same steps but now we set L and the privacy parameter (and consequently the distortion rate) of the planar Laplace as follows:

$$L = 173\text{m} \qquad \epsilon = \frac{\ln 2}{60} \qquad \mathbb{L}[Z \mid W] \approx 173.12\text{m} \tag{4.36}$$

Similarly to the previous section, training G with the cross entropy fails to converge, producing generators that achieve no privacy protection. As a consequence, we only show the results of training G with respect to mutual information.

The result of the application of the Laplace mechanism is illustrated in fig. 4.4 (leftmost plot). The empirical distortion is ≈ 172.35m.

Following the same pattern as in Section section 4.4.1, we train G and C. The training of G is performed for 30 epochs during each iteration with a batch size of 512 samples and a learning rate of 0.0001.

The classifier C is trained for 3000 epochs with a batch size of 512 samples and 0.001 as the value for the learning rate during each iteration. We are particularly interested in the 24th iteration where C's performance is degraded by the obfuscation performed by G trained during the previous iteration. Training C with 32 samples batch size and learning rate set to 0.001 for 100 epochs with the obfuscated data gives the results reported in table 4.7. In

Data	Accuracy	F1_score
Training data	≈ 0.55	≈ 0.54
Validation data	≈ 0.53	≈ 0.53
Test data	≈ 0.52	≈ 0.51

Table 4.7 – Summary of the experiment with the the noise produced by the proposed method.

this case, increasing the number of epochs does not improve the classification precision and makes C more prone to over-fitting.

The obfuscation provided by G at the 24th iteration produces the distributions on the testing illustrated in fig. 4.4 (rightmost plot). The empirical distortion is ≈ 171.50m. The estimated Bayes error for the two mechanisms is reported in table 4.6.

4.5 Experiments on the Gowalla dataset

In the previous section we saw that our method behaves as expected in a simple synthetic dataset, producing an obfuscation mechanism that is close to the optimal one (when G is trained wrt mutual information). We now study the behaviour of our method to real location data from the Gowalla dataset. Since cross entropy was shown to be unsound, we only present results using mutual information for training G.

The dataset

The dataset consists of data extracted from the *Gowalla* dataset [LK], a collection of check-ins made available by the Gowalla location-based social network. Among all the provided features, only the users' identifiers (classes), the latitude and longitude of the check-in locations are considered.

The data is selected as follows:

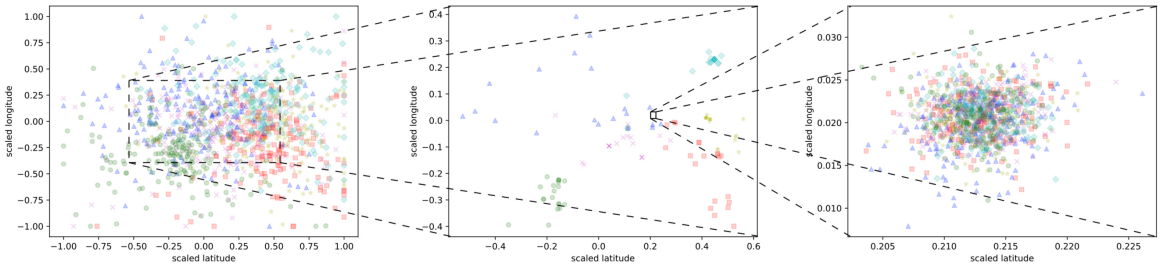


Figure 4.5 – Gowalla testing data. From left to right: Laplace noise, no noise, our noise produced using mutual information. $L = 1150$m.

1. we consider a squared region centered in 5, Boulevard de Sébastopol, Paris, France with 4500m long side;
2. we select the 6 users who checked in the region most frequently, we retain their locations and discard the rest;
3. we filter the obtained locations to reduce the overlapping of the data belonging to different classes by randomly selecting for each class 82 location samples for training and validation purpose, and 20 samples for the test.

We obtain 492 pairs $(locations, id)$ to train and validate the model, and 120 to test it. For each of these, the generator creates 10 pairs with noisy locations using different seeds. As usual, G_0 does it using the Laplace function, the other G_i's use the mechanism learnt at the previous step $i - 1$. Thus in total we obtain 4920 pairs for training and 1200 for testing. Figure 4.5 shows the result of the mechanism applied to the testing data, where each color corresponds to a different user.

Number of cells			
13×13	65×65	130×130	260×260
0.12	0.06	0.04	0.03

(a) Training data.

Number of cells			
13×13	65×65	130×130	260×260
0.11	0.04	0.03	0.03

(b) Testing data.

Table 4.8 – Estimation of $B(X \mid Z)$ on the original version of the data from Gowalla.

4.5.1 Experiment 3: relaxed utility constraint

In this experiment we study the case of a large upper bound on the utility loss, which would potentially allow to achieve the maximum utility. We set L and the privacy parameter (and

	Number of cells							
	13 × 13		65 × 65		130 × 130		260 × 260	
Obf	Lap	Our	Lap	Our	Lap	Our	Lap	Our
10	0.56	0.83	0.37	0.83	0.19	0.82	0.06	0.80
100	0.57	0.83	0.53	0.83	0.46	0.82	0.31	0.81
200	0.57	0.83	0.55	0.83	0.50	0.82	0.40	0.81
500	0.57	0.83	0.56	0.83	0.54	0.82	0.48	0.81

(a) Training data.

	Number of cells							
	13 × 13		65 × 65		130 × 130		260 × 260	
Obf	Lap	Our	Lap	Our	Lap	Our	Lap	Our
10	0.51	0.83	0.18	0.82	0.07	0.80	0.01	0.79
100	0.56	0.83	0.45	0.82	0.30	0.81	0.13	0.80
200	0.56	0.83	0.49	0.82	0.38	0.81	0.21	0.80
500	0.56	0.83	0.53	0.82	0.46	0.81	0.33	0.80

(b) Testing data.

Table 4.9 – Estimation of $B(X \mid Z)$ on the Gowalla data for the Laplace and for our mechanisms, with $L = 1150$m. The utility loss for training and testing data is $\approx 1127.83\text{m} - 1132.63\text{m}$ respectively for the Laplace and $\approx 961.38\text{m} - 979.40\text{m}$ for ours. The optimal mechanism gives $B(X \mid Z) = 1 - 1/6 = 0.83$.

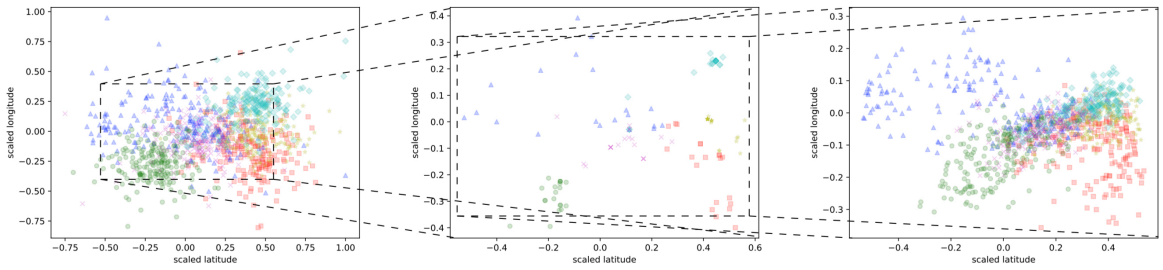


Figure 4.6 – Gowalla testing data. From left to right: Laplace noise, no noise, our noise produced using mutual information. $L = 518$m.

consequently $\mathbb{L}[Z \mid W]$) of the planar Laplace as follows:

$$L = 1150\text{m} \qquad \epsilon = \frac{\ln 2}{400} \qquad \mathbb{L}[Z \mid W] \approx 1154.15\text{m} \tag{4.37}$$

The results for the Laplace and our method are illustrated in fig. 4.5. As we can see, the utility constraint is relaxed enough to allow our method to achieve the maximum privacy. As reported in table 4.9, indeed, the Bayes error is close to that of random guess, namely $1 - 1/6 \approx 0.83$. Table 4.8 shows the part of the Bayes error due to the discretization of the domain $\mathcal{Z}$. The planar Laplace, on the other hand, confirms the relatively limited level of privacy as observed in the synthetic data.

	Number of cells							
	13 × 13		65 × 65		130 × 130		260 × 260	
Obf	Lap	Our	Lap	Our	Lap	Our	Lap	Our
10	0.38	0.50	0.29	0.42	0.20	0.37	0.27	0.08
100	0.39	0.51	0.36	0.44	0.34	0.43	0.27	0.40
200	0.39	0.51	0.36	0.44	0.35	0.43	0.31	0.41
500	0.38	0.51	0.37	0.44	0.36	0.43	0.34	0.42

(a) Training data.

	Number of cells							
	13 × 13		65 × 65		130 × 130		260 × 260	
Obf	Lap	Our	Lap	Our	Lap	Our	Lap	Our
10	0.34	0.47	0.20	0.36	0.08	0.35	0.03	0.12
100	0.37	0.49	0.32	0.41	0.25	0.38	0.15	0.32
200	0.37	0.48	0.33	0.41	0.30	0.39	0.21	0.35
500	0.37	0.49	0.35	0.42	0.32	0.40	0.28	0.38

(b) Testing data.

Table 4.10 – Estimation of $B(X \mid Z)$ on the Gowalla data for the Laplace and for our mechanisms, with $L = 518$m. The utility loss fort training and testing data is $\approx 523.40\text{m} - 535.21\text{m}$ for the Laplace and $\approx 487.34\text{m} - 502.89\text{m}$ for ours. We could not compute the optimal mechanism.

4.5.2 Experiment 4: stricter utility constraint

We consider now a much tighter utility constraint, and we set the parameters of the planar Laplace as follows:

$$L = 518\text{m} \qquad \epsilon = \frac{\ln 2}{180} \qquad \mathbb{L}[Z \mid W] \approx 519.37\text{m} \tag{4.38}$$

The results of the application of the Laplace and of our method to the testing data are shown in fig. 4.6, and the Bayes error is reported in table 4.10. As the grid becomes finer, both the planar Laplace and our method become more sensitive to the number of samples, in the sense that the (approximation of) the Bayes error grows considerably as the number of samples increases. This is not surprising: when the cells are small they tend to have a limited number of hits. Therefore the number of hits whose class is in minority (in a given cell), and hence not selected as the best guess, is limited. Note that these minority hits are those that contribute to the Bayes error.

4.6 Final remarks

Related work

For a detailed discussion of the the literature concerning the application of linear programming and adversarial ML to the problem of building the optimal obfuscation mechanisms which guarantee a trade-off between privacy and utility, we refer to section 1.3. However we would like to add a few remarks on the most important among the works that inspired ours. One important contribution on the topic of the application of adversarial networks to privacy-protection mechanisms design have been also proposed by the authors of [TWI17, HKC+17]. They have developed a theoretical framework similar to ours. From the methodological point of view the main difference is that in the implementation they use as target function the cross entropy rather than the mutual information. Hence in our setting the convergence of their method may be problematic, due to the "swapping effect" described in example 1.

One of the side contributions of this work is a method to compute mutual information in neural network (cfr. section 4.3). Recently, Belghazi et al. have proposed MINE, an efficient method to neural estimation of mutual information [BBR+18], inspired by the framework of [NCT16] for the estimation of a general class of functions which can be represented as f-divergences. These methods work also in the continuous case and for high-dimensional data.

In our case, however, we are dealing with a discrete domain, and we can compute directly and *exactly* the mutual information. Another reason for developing our own method is that we need to deal with a loss function that contains not only the mutual information, but also a component representing utility, and depending on the notion of utility the result may not be an f-divergence.

Summary

We have proposed an approach based on adversarial nets to generate obfuscation mechanisms with a good privacy-utility trade-off. The advantage of our method is twofold: with respect to the linear programming methods, we can work on a continuous domain instead of a small grid; with respect to the analytic methods (such as the Planar Laplace mechanism) our approach is data-driven, taking into account prior knowledge about the users. However we recall that strong privacy properties such as differential privacy (and derived notions, e.g. geo-indistinguishability) are not guaranteed by our approach and the comparison to linear programming based solutions mainly concerns the performances. In fact, we remind that under certain hypothesis, guarantees of (universal) optimality can be proven by modeling the privacy mechanism design as a linear programming optimization problem [KM17, KM18].

Although our approach is inspired by the GAN paradigm, it departs significantly from it: in our case, the distribution has to be "invented" rather than "imitated". Hence we need different techniques for evaluating a distribution. To achieve our goal, we propose a new

method based on the mutual information between the supervised and the predicted class labels.

We explain that the use of the use of mutual information (instead of the cross entropy) for the generator is crucial for convergence. On the other hand for the classifier it is possible to use cross entropy and it is more efficient.

We evaluate the obfuscation mechanism produced by our method on real location data from the Gowalla dataset. We compare our mechanism with the planar Laplace [ABCP13] and with the optimal one, when it is possible to compute or determine theoretically the latter.

We show that, taking into account the different notions of privacy embodied by our proposed mechanism and the Laplace method, the performance of the former is better than the that of the latter, and, also, not so far from the optimal. We have made publicly available the implementation and the experiments at https://gitlab.com/MIPAN/mipan.

So far we have analyzed how ML techniques and notions from information theory can help us deal with leakage estimation and privacy protection mechanism design. In the next chapter we focus on the study of a particular notion of entropy from the information theory field, i.e. the Rényi min-entropy, and how it can help improve, at least under certain conditions, the solutions to the problem of dimensionality reduction via feature selection in ML.

CHAPTER 5

Feature selection in machine learning: Rényi min-entropy vs Shannon entropy

In this chapter, we consider the typical scenario of classification problems and we propose an algorithm to sort features from those which are the most informative with respect to the class (label) to those which are less informative, as introduced in the final paragraphs of section 1.2. We address the problem from the data dimensionality reduction standpoint. Among the works concerned with this topic, we cite [JDM00, GE03, LY05, BPZL12, VE14, BHS15, SSGC17, Nak18, CLWY18, LLWW18]. The identification of the "best" features for classification is indeed very helpful in order to avoid too high a training complexity and, in many cases, it has been shown to improve the accuracy of the classification. However, we claim that a similar reasoning can be applied to fields of privacy and security as well. In fact, if we consider the classes as what we would like to keep secret, knowing which features leak more information about them can help to understand what part of the data a privacy protection mechanism should mainly defend against exposure.

The known methods for reducing the dimensionality can be divided in two categories: those which transform the feature space by reshaping the original features into new ones (*feature extraction*), and those which select a subset of the features (*feature selection*). The second category can in turn be divided in three groups: *wrapper*, *embedded*, and *filter* methods. The last group has the advantage of being classifier-independent, more robust with respect to the risk of over-fitting, and more amenable to a principled approach. In particular, several proposals for feature selection have successfully applied concepts and techniques from information theory [Bat94, YM99, Fle04, PLD05, BPZL12, VE14, BHS15].

The underlining idea is that the smaller the conditional (aka residual) entropy of the classes given a certain set of features is, the more likely the classification of a sample is to be correct. Finding a good set of features corresponds therefore to identifying a subset of the initial one, as small as possible, for which such conditional entropy is below a certain threshold. More formally, the problem of feature selection can be stated as follows: given the random variables F and C, modeling respectively a set of *features* and a set of *classes*[1],

[1]When clear from the context, we will use the same notation to represent both the random variable and its

find a minimum-size subset $S \subseteq F$ such that the conditional entropy of C given S is below a certain threshold. Namely:

$$S = \underset{S'}{\operatorname{argmin}} \{|S'| \mid S' \subseteq F \text{ and } H(C \mid S') \leqslant h\} \tag{5.1}$$

where h is the given threshold, $|S'|$ is the number of elements of S', and $H(C \mid S')$ is the conditional Shannon entropy of C given F.

All the information-theoretic approaches to feature selection that have been proposed are based, as far as we know, on Shannon entropy, with the notable exception of [EK13] that considered the Rényi entropies H_α, where α is a parameter ranging over all the positive reals and ∞ (cfr. section 2.2 for a detailed analysis). We remind that the Shannon entropy is represented by the symbols H_1. We use H instead when the context does not require for the parameter value $\alpha = 1$ to be specified.

In this work, we develop an approach to feature selection based on the particular case of the Rényi min-entropy H_∞, and we compare it with the one based on Shannon entropy. Our approach and analysis actually depart significantly from [EK13]; the differences between their solution and the one that we propose will be explained in section 5.4.

The starting point for an approach based on the min-entropy is to use the conditional min-entropy in eq. (5.1). As already discussed in section 2.2, we adopt the conditional min-entropy definition introduced in [Smi09]. More specifically, this notion captures the (converse of) the probability of error of a rational attacker who knows the probability distributions and tries to infer a secret from some correlated observables. "Rational" here means that the attacker will try to minimize the expected probability of error, by selecting the secret with highest posterior probability. Note the similarity with the classification problem, where the decisions are made by choosing a class on the basis of the observable features, trying to minimize the expected probability of classification error (misclassification). It is therefore natural to investigate the potentiality of this notion in the context of feature selection. Moreover, it is worth noticing that, since we assume that the attacker is rational and knows the probability distributions, we are considering the Bayes attacker and, correspondingly, the classifier is the (ideal) Bayes classifier.

By replacing the Shannon entropy H with the Rényi min-entropy H_∞, the problem described in eq. (5.1) becomes:

$$S = \underset{S'}{\operatorname{argmin}} \{|S'| \mid S' \subseteq F \text{ and } H_\infty(C \mid S') \leqslant h\} \tag{5.2}$$

where $H_\infty(C \mid S')$ is the conditional min-entropy of C given F. Because of the correspondence between $H_\infty(\cdot \mid \cdot)$ and the Bayes error, we can interpret eq. (5.2) as stating that S is the minimal set of features for which the (ideal) Bayes classifier achieves the desired level of accuracy.

---

supporting set.

5.1 Problem definition

In this section we state formally the problem of finding a minimal set of features that satisfies a given bound on the classification's accuracy, and then we show that the problem is NP-hard. More precisely, we are interested in finding a minimal set with respect to which the posterior Rényi min-entropy of the classification is bounded by a given value. We recall that the posterior Rényi min-entropy is equivalent to the Bayes classification error.

The corresponding problem for Shannon entropy is well studied in the literature of feature selection, and its NP-hardness is a well known theorem in the area. However for the sake of comparing it with the case of Rényi min-entropy, we restate it here in the same terms as for the latter.

In the following, F stands for the set of all features, and C for the random variable that takes value in the set of classes.

Definition 4 (MIN-SET)**:** Let h be a non-negative real. The minimal-set problems for Shannon entropy and for Rényi min-entropy are defined as the problems of determining the set of features S such that

Shannon	MIN-SET$_1$	$S = \operatorname{argmin}_{S'} \{\|S'\| \mid S' \subseteq F \text{ and } H_1(C \mid S') \leqslant h\},$
Rényi	MIN-SET$_\infty$	$S = \operatorname{argmin}_{S'} \{\|S'\| \mid S' \subseteq F \text{ and } H_\infty(C \mid S') \leqslant h\}.$

We now show that the above problems are NP-hard

Theorem 4: Both MIN-SET$_1$ and MIN-SET$_\infty$ are NP-hard.

Proof. Consider the following decision problem MIN-FEATURES: Let M be a set of examples, each of which is composed of a a binary value specifying the value of the class and a vector of binary values specifying the values of the features. Given a number n, determine whether or not there exists some feature set S such that:

- S is a subset of the set of all input features.
- S has cardinality n.
- There exists no two examples in M that have identical values for all the features in S but have different class values.

In [DR94] it is shown that MIN-FEATURES is NP-hard by reducing to it the VERTEX-COVER problem, which is known to be NP-complete [Kar72]. We recall that the VERTEX-COVER problem problem may be stated as the following question: given a graph G with vertices V and edges E, is there a subset V' of V, of size m, such that each edge in E is connected to at least one vertex in V'?

To complete the proof, it is sufficient to show that we can reduce MIN-FEATURES to MIN-SET$_1$ and MIN-SET$_\infty$. Set $h = 0$, and let

$$S = \operatorname{argmin}_{S'} \{|S'| \mid S' \subseteq F \text{ and } H_\alpha(C \mid S') = 0\},$$

where $\alpha = 1$ or $\alpha = \infty$. Note that for both values of α, $H_\alpha(C \mid S) = 0$ means that the uncertainty about C is 0 once we know the value of all features in S, and this is possible only if there exists no two examples in that have identical values for all the features in S but have different class values. Hence to answer MIN-FEATURES it is sufficient to check whether $|S| \leqslant m$ or $|S| > m$. □

Given that the problem is NP-hard, there is no "efficient" algorithm (unless P = NP) for computing exactly the minimal set of features S satisfying the bound on the accuracy. It is however possible to compute efficiently an approximation of it, as we will see in next section, where we propose a linear "greedy" algorithm which computes an approximation of the minimal S.

5.2 Proposed algorithm

Let F be the set of features at our disposal, and let C be random variable ranging on the set of classes. Our algorithm is based on forward feature selection and dependency maximization: it constructs a monotonically increasing sequence $\{S^t\}_{t\geqslant 0}$ of subsets of F, and, at each step, the subset S^{t+1} is obtained from S^t by adding the next feature in order of importance (i.e., the informative contribution to classification), taking into account the information already provided by S^t. The measure of the "order of importance" is based on conditional min-entropy. The construction of the sequence is assumed to be done interactively with a test on the accuracy achieved by the current subset, using one or more classifiers. This test will provide the stopping condition: once we obtain the desired level of accuracy, the algorithm stops and gives as result the current subset S^T. Of course, achieving a level of accuracy $1-\varepsilon$ is only possible if $\mathcal{B}(C \mid F) \leqslant \varepsilon$.

Definition 5: The series $\{S^t\}_{t\geqslant 0}$ and $\{f^t\}_{t\geqslant 1}$ are inductively defined as follows:

$$\begin{array}{rcl} S^0 & \stackrel{\text{def}}{=} & \varnothing \\ f^{t+1} & \stackrel{\text{def}}{=} & \operatorname{argmin}_{f\in F\setminus S^t} H_\infty(C \mid f, S^t) \\ S^{t+1} & \stackrel{\text{def}}{=} & S^t \cup \{f^{t+1}\} \end{array}$$

The algorithms in [BPZL12] and [VE14] are analogous, except that they use Shannon entropy. They also define f^{t+1} based on the maximization of mutual information instead of the minimization of conditional entropy, but this is irrelevant. In fact $I_1(C; f \mid S^t) = H_1(C \mid S^t) - H_1(C \mid f, S^t)$, hence maximizing $I_1(C; f \mid S^t)$ with respect to f is the same as minimizing $H_1(C \mid f, S^t)$ with respect to f.

Our algorithm is locally optimal, in the sense stated by the following proposition.

Proposition 9: At every step, the set S^{t+1} minimizes the Bayes error of the classification among those which are of the form $S^t \cup \{f\}$, namely:

$$\forall f \in F \;\; \mathcal{B}(C \mid S^{t+1}) \leqslant \mathcal{B}(C \mid S^t \cup \{f\})$$

Proof. Let $\vec{v}, v, v'$ represent generic value tuples and values of S^t, f and f^{t+1}, respectively. Let c represent the generic value of C. By definition, $H_\infty(C \mid S^{t+1}) \leqslant H_\infty(C \mid S^t \cup \{f\})$, for every $f \in F$. From eq. (2.10) we then obtain

$$\sum_{\vec{v},v} \max_c(p(\vec{v}, v|c)p(c)) \leqslant \sum_{\vec{v},v'} \max_c(p(\vec{v}, v'|c)p(c))$$

Using the Bayes theorem section 2.2, we get

$$\sum_{\vec{v},v} p(\vec{v}, v) \max_c p(c|\vec{v}) \leqslant \sum_{\vec{v},v'} p(\vec{v}, v') \max_c p(\vec{v}, v'|c)$$

Then, from the definition eq. (2.11) we deduce

$$\mathcal{B}(C \mid S^t \cup \{f^{t+1}\}) \leqslant \mathcal{B}(C \mid S^t \cup \{f\})$$

□

In the following sections we analyze some extended examples to illustrate how the algorithm works, and also compare it with the ones of [BPZL12] and [VE14].

5.2.1 An example in which Rényi min-entropy gives a better feature selection than Shannon entropy

Let us consider the dataset in table 5.1, containing ten records labeled each by a different class, and characterized by six features (columns $f_1, \ldots, f_5$). We note that f_0 separates the classes in two sets of four and six elements respectively, while all the other columns are characterized by having two values, each of which univocally identify one class, while the third value is associated to all the remaining classes. For instance, in column f_1 value A univocally identifies the record of class 0, B univocally identifies the record of class 1, and all the other records have the same value along that column, i.e. C.

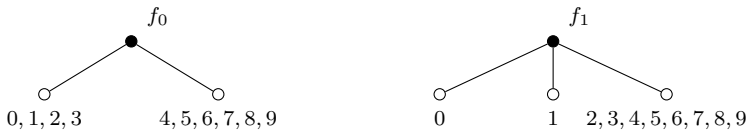
The last five features combined are necessary and sufficient to completely identify all classes, without the need of the first one. Note that the last five features can be replaced by f_0 for this purpose. In fact, each pair of records which are separated by one of the features $f_1, \ldots, f_5$, have the same value in column f_0.

If we apply the discussed feature selection method and we look for the feature that minimizes $H(C|f_i)$ for $i \in \{0, \ldots, 5\}$ we obtain that:

- The first feature selected with Shannon is f_0, in fact $H_1(C|f_0) \approx 2.35$ and $H_1(C|f_{\neq 0}) = 2.4$. (The notation $f_{\neq 0}$ stands for any of the f_i's except f_0.) In general, indeed, with Shannon entropy the method tends to choose a feature which splits the Classes in a way as balanced as possible. The situation after the selection of the feature f_0 is shown in fig. 5.1(a).

Class	f_0	f_1	f_2	f_3	f_4	f_5
0	A	C	F	I	L	O
1	A	D	F	I	L	O
2	A	E	G	I	L	O
3	A	E	H	I	L	O
4	B	E	F	J	L	O
5	B	E	F	K	L	O
6	B	E	F	I	M	O
7	B	E	F	I	N	O
8	B	E	F	I	L	P
9	B	E	F	I	L	Q

Table 5.1 – The dataset for the example in section 5.2.1.



(a) Selection with Shannon. (b) Selection with Rényi.

Figure 5.1 – Classes separation after the selection of the first feature.

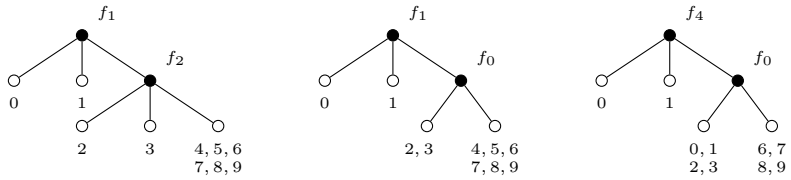


Figure 5.2 – Selection of the second feature with Rényi.

- The first feature selected with Rényi min-entropy is either f_1 or f_2 or f_3 or f_4 or f_5, in fact $H_\infty(C|f_0) \approx 2.32$ and $H_\infty(C|f_{\neq 0}) \approx 1.74$. In general, indeed, with Rényi min-entropy the method tends to choose a feature which divides the classes in as many sets as possible. The situation after the selection of f_1 is shown in fig. 5.1(b).

Going ahead with the algorithm, with Shannon entropy we will select one by one all the other features, and as already discussed we will need all of them to completely identify all classes. Hence at the end the method with Shannon entropy will return all the six features (to achieve perfect classification). On the other hand, with Rényi min entropy we will select all the remaining features except f_0 to obtain the perfect discrimination. In fact, at any stage the selection of f_0 would allow to split the remaining classes in at most two sets, while any

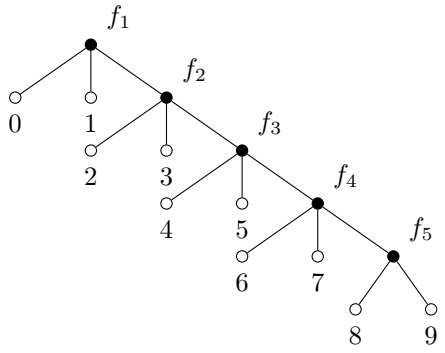


Figure 5.3 – Sequence of class splitting with Rényi.

other feature not yet considered will split the remaining classes in three sets. As already hinted, with Rényi we choose the feature that allows to split the remaining classes in the highest number of sets, hence we never select f_0. For instance, if we have already selected f_1, we have $H_\infty(C|f_1 f_0) \approx 1.32$ while $H_\infty(C|f_1 f_{\neq 0}) = 1$. If we have already selected f_4, we have $H_\infty(C|f_4 f_0) \approx 1.32$ while $H_\infty(C|f_4 f_{\neq 0}) = 1$. See fig. 5.2.

At the end, the selection of features using Rényi entropy will determine the progressive splitting represented in fig. 5.3. The order of selection is not important: this particular example is conceived so that the features $f_1, \ldots, f_5$ can be selected in any order, the residual entropy is always the same.

Discussion It is easy to see that, in this example, the algorithm based on Rényi min-entropy gives a better result not only at the end, but also at each step of the process. Namely, at step t (cfr. definition 5) the set S^t of features selected with Rényi min-entropy gives a better classification (i.e., more accurate) than the set S'^t that would be selected using Shannon entropy. More precisely, we have $\mathcal{B}(C \mid S^t) < \mathcal{B}(C \mid S'^t)$. In fact, as discussed above the set S'^t contains necessarily the feature f_0, while S^t does not. Let S^{t-1} be the set of features selected at previous step with Rényi min-entropy, and f^t the feature selected at step t (namely, $S^{t-1} = S^t \backslash \{f^t\}$). As argued above, the order of selection of the features $f_1, \ldots, f_5$ is irrelevant, hence we have $\mathcal{B}(C \mid S^{t-1}) = \mathcal{B}(C \mid S'^t \backslash \{f_0\})$ and the algorithm *could* equivalently have selected $S'^t \backslash \{f_0\}$. The next feature to be selected, with Rényi, must be different from f_0. Hence by proposition 9, and by the fact that the order of selection of $f_1, \ldots, f_5$ is irrelevant, we have: $\mathcal{B}(C \mid S^t) = \mathcal{B}(C \mid (S'^t \backslash \{f_0\}) \cup \{f^t\}) < \mathcal{B}(C \mid S'^t)$.

As a general observation, we can see that the method with Shannon tends to select the feature that divides the classes in sets (one for each value of the feature) as balanced as possible, while our method tends to select the feature that divides the classes in as many sets as possible, regardless of the sets being balanced or not. In general, both Shannon-based and Rényi-based methods try to minimize the height of the tree representing the process of the splitting of the classes, but the first does it by trying to produce a tree *as balanced as possible*, while the second one tries to do it by producing a tree *as wide as possible*. Which of the method is best, it depends on the correlation of the features. Shannon works better

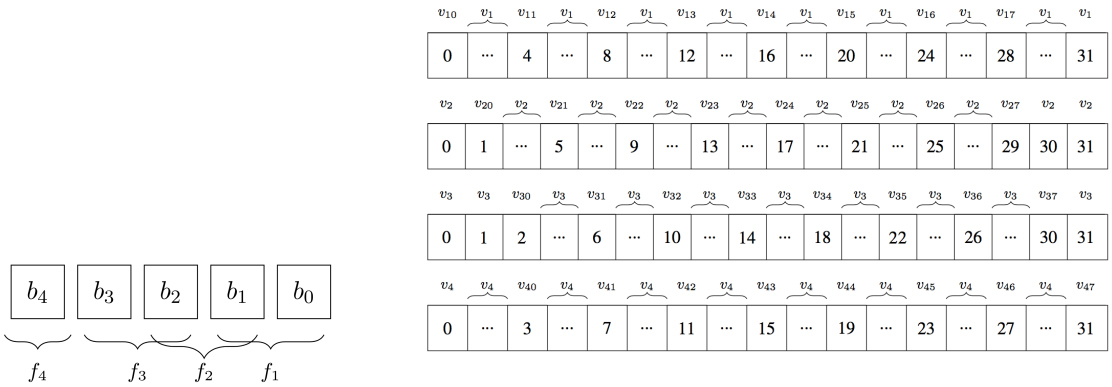


Figure 5.4 – Features F (left) and F' (right).

when there are enough uncorrelated (or not much correlated) features, so that the tree can be kept balanced while being constructed. Next section shows an example of such situation. Rényi, on the contrary, is not so sensitive to correlation and can work well also when the features are highly correlated, as it was the case in the example of this section.

The experimental results in section 5.3 show that, at least in the cases we have considered, our method outperforms the one based on Shannon entropy. In general however the two methods are incomparable, and perhaps a good practice would be to construct both sequences at the same time, so to obtain the best result of the two.

5.2.2 An example in which Shannon entropy may give a better feature selection than Rényi min-entropy

Consider a dataset containing samples equally distributed among 32 classes, indexed from 0 to 31. Assume that the data have 8 features divided in 2 types F and F', each of which consisting of 4 features: $F = \{f_1, f_2, f_3, f_4\}$ and $F' = \{f'_1, f'_2, f'_3, f'_4\}$. The relation between the features and the classes is represented in fig. 5.4.

Assume that the features of type F, and their relation with C are as follows: Given the binary representation of the classes $b_4b_3b_2b_1b_0$, f_1 consists of the bits b_1b_0, f_2 consists of the bits b_2b_1, f_3 consists of the bits b_3b_2, and f_4 consists of the bit b_4 only. All the v_{ij} values allow to identify exactly one class. Concerning the features of type F', assume that each of them can range in a set of 9 values, and that these sets are mutually disjoint. We use the following notation: for $1 \leqslant i \leqslant 4 \quad f'_i = \{v_i\} \cup \{v_{ij} \mid 0 \leqslant j \leqslant 7\}$. For instance, if in a certain sample the f'_1 value is v_{10}, it means that the sample can be univocally classified as belonging to class 0. If the f'_2 value is v_{22}, it belongs to class 9, etc. As for the v_i's, each of them is associated to a set of 24 classes, with uniform distribution. For instance, the value v_1 is associated to classes $1, 2, 3, 5, \ldots, 29, 30, 31$. The situation is represented in fig. 5.4.

Let us select the first feature f in order of importance for the classification. We will

consider at the same time our method, based on Rényi min-entropy, and the method of [BPZL12] and [VE14]. In both cases, the aim is to choose f such that it maximizes the mutual information $I_\alpha = H_\alpha(C) - H_\alpha(C \mid f)$, or equivalently, that minimizes the conditional entropy $H_\alpha(C \mid f)$, where α is the entropy parameter ($\alpha = 1$ for Shannon, $\alpha = \infty$ for Rényi min-entropy).

For the features of type F we have:

$$H_1(C \mid f_1) = H_1(C \mid f_2) = H_1(C \mid f_3) = 3$$

while

$$H_1(C \mid f_4) = 4 > 3$$

On the other hand, with respect to the features of type F', we have

$$H_1(C \mid f'_1) = H_1(C \mid f'_2) = H_1(C \mid f'_3) = H_1(C \mid f'_4) \approx 3.439 > 3$$

So, the first feature selected using Shannon entropy would be f_1, f_2 or f_3. Let us assume that we pick f_1. Hence with Shannon the first set of the series is $S_1^1 = \{f_1\}$.
Let us now consider our method based on Rényi min-entropy. For the F feature the conditional entropy is the same as for Shannon:

$$H_\infty(C \mid f_1) = H_\infty(C \mid f_2) = H_\infty(C \mid f_3) = 3$$

$$H_\infty(C \mid f_4) = 4 > 3$$

But for the F' features Rényi min-entropy gives a different value. In fact we get:

$$H_\infty(C \mid f'_1) = H_\infty(C \mid f'_2) = H_\infty(C \mid f'_3) = H_\infty(C \mid f'_4) \approx 1.83 < 2$$

So, the first feature selected using Rényi min entropy would be f'_1, f'_2, f'_3, or f'_4. Let us assume that we pick f'_1. Hence with our method the first set of the series is $S_\infty^1 = \{f'_1\}$.

For the selection of the second feature, we have

$$H_1(C \mid f_1, f_2) = H_1(C \mid f_1, f_4) = 2$$

while

$$H_1(C \mid f_1, f_3) = 1$$

$$H_1(C \mid f_1, f'_1) = H_1(C \mid f_1, f'_2) = H_1(C \mid f_1, f'_3) = H_1(C \mid f_1, f'_4) > 2$$

Hence the second feature with Shannon can only be f_3, and thus the second set in the sequence is $S_1^2 = \{f_1, f_3\}$.

With Rényi min-entropy we have:

$$H_\infty(C \mid f'_1, f_4) > H_\infty(C \mid f'_1, f_i) \text{ for } i = 1, 2, 3$$

$$H_\infty(C \mid f'_1, f_i) > H_\infty(C \mid f'_1, f'_j) \text{ for } i = 1, 2, 3 \text{ and } j = 2, 3, 4$$

Hence with our method the second feature will be f'_2, f'_3, or f'_4. Let us assume that we choose f'_2. Thus the second set of the series is $S^2_\infty = \{f'_1, f'_2\}$. At step 3 one of the possible outcomes of the algorithm based on Shannon is the set of features $S^3_1 = \{f_1, f_3, f_4\}$, and one of the possible outcomes of the algorithm based on Rényi is $S^3_\infty = \{f'_1, f'_2, f'_i\}$ where i can be, equivalently, 3 or 4. At this point the method with Shannon can stop, since the residual Shannon entropy of the classification is $H_1(C \mid S^3_1) = 0$, and also the Bayes risk is $\mathcal{B}(C \mid S^3_1) = 0$, which is the optimal situation in the sense that the classification is completely accurate. S^3_∞ on the contrary does not contain enough features to give a completely accurate classification, for that we have to make a further step. We can see that $S^4_\infty = F'$, and finally we have $H_\infty(C \mid S^4_\infty) = 0$.

5.3 Evaluation

In this section we evaluate the method for feature selection that we have proposed, and we compare it with the one based on Shannon entropy by [BPZL12] and [VE14].

To evaluate the effect of feature selection, some classification methods have to be trained and tested on the selected data. We used two different methods to avoid the dependency of the result on a particular algorithm. We chose two widely used classifiers: the Support Vector Machines (SVM) and the Artificial Neural Networks (ANN).

Even though the two methods are very different, they have in common that their efficiency is highly dependent on the choice of certain parameters. Therefore, it is worth spending some effort to identify the best values. Furthermore, we should take into account that the particular paradigm of SVM we chose only needs two parameters to be set, while for ANN the number of parameters increases (at least four).

It is very important to choose values as robust as possible for the parameters. It goes without saying that the strategy used to pick the best parameter setting should be the same for both Shannon entropy and Rényi min-entropy. On the other hand for SVM and ANN we used two different hyper-parameter tuning algorithms, given that the number and the nature of the parameters to be tuned for those classifiers is different.

In the case of SVM we tuned the cost parameter of the objective function for margin maximization (*C-SVM*) and the parameter which models the shape of the RBF kernel's bell curve (γ). Grid-search and Random-search are quite time demanding algorithms for the hyper-parameter tuning task but they're also widely used and referenced in literature when it comes to SVM. Following the guidelines in [CL11] and [Ped11], we decided to use Grid-search, which is quite suitable when we have to deal with only two parameters. In particular we performed Grid-search including a 10 folds CV step.

Things are different with ANN because many more parameters are involved and some of them change the topology of the network itself. Among the various strategies to attack this problem we picked Bayesian Optimization [SLA12]. This algorithm combines steps of extensive search for a limited number of settings before inferring via Gaussian Processes (GP) which is the best setting to try next (with respect to the mean and variance and compared to

the best result obtained in the last iteration of the algorithm). In particular we tried to fit the best model by optimizing the following parameters:

- number of hidden layers
- number of hidden neurons in each layer
- learning rate for the gradient descent algorithm
- size of batches to update the weight on network connections
- number of learning epochs

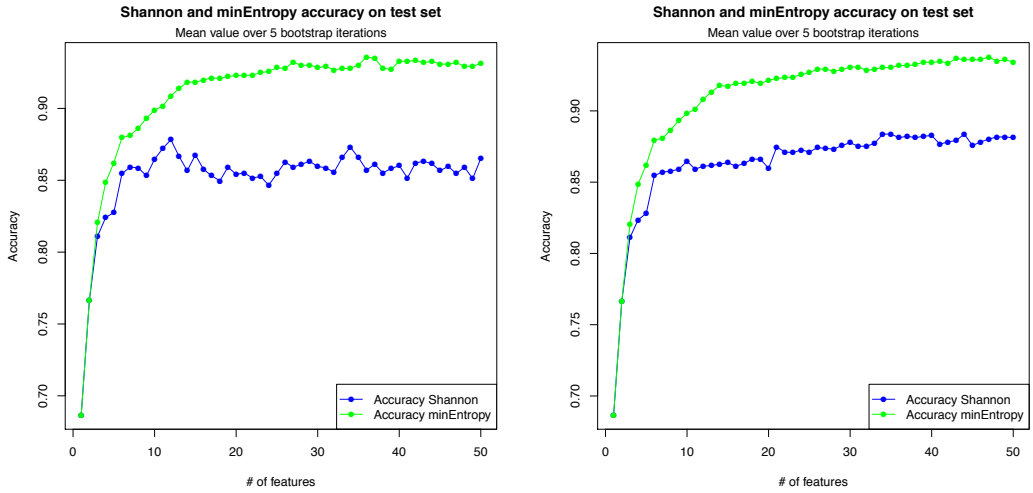


Figure 5.5 – Accuracy of the ANN and SVM classifiers on the Basehock dataset.

To this purpose, we included in the pipeline of our code the *Spearmint* Bayesian optimization codebase. Spearmint, whose theoretical bases are explained in [SLA12], calls repeatedly an objective function to be optimized. In our case the objective function contained some *tensorflow* machine learning code which run a 10 folds CV over a dataset and the objective was to maximize the accuracy of validation. The idea was to obtain a model able to generalize as much as possible using only the selected features before testing on a dataset which had never been seen before.

We had to decide the stopping criterion, which is not provided by *Spearmint* itself. For the sake of simplicity we decided to run it for a time lapse which has empirically been proven to be sufficient in order to obtain results meaningful for comparison. A possible improvement would be to keep running the same test (with the same number of features) for a certain amount of time without resetting the computation history of the package and only stop testing a particular configuration if the same results is output as the best for k iterations in a row (for a given k).

Another factor, not directly connected to the different performances obtained with different entropies, but which is important for the optimization of ANN, is the choice of the activation functions for the layers of neurons. In our work we have used ReLU for all layers because it is well known that it works well for this aim, it is easy to compute (the only operation involved is the max) and it avoids the sigmoid saturation issue.

Experiments

As already stated, at the i-th step of the feature selection algorithm we consider all the features which have already been selected in the previous $i-1$ step(s). For the sake of limiting the execution time, we decided to consider only the first 50 selected features with both metrics. We tried our pipeline on the following datasets:

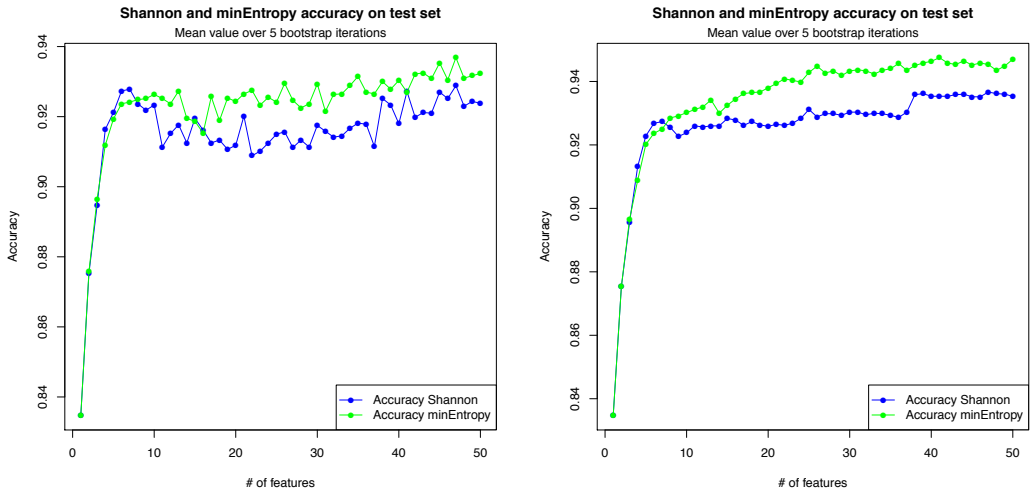


Figure 5.6 – Accuracy of the ANN and SVM classifiers on the Gisette dataset.

- Basehock dataset: 1993 instances, 4862 features, 2 classes. This dataset has been obtained from the 20 newsgroup original dataset.
- Semeion dataset: 1593 instances, 256 features, 10 classes. This is a dataset with encoding of hand written characters.
- Gisette dataset: 6000 instances, 5000 features, 2 classes. This is the discretized version of the NIPS 2003 dataset which can be downloaded from the site of Professor Gavin Brown, Manchester University.

We implemented a bootstrap procedure (5 iterations on each dataset) to shuffle data and make sure that the results do not depend on the particular split between training, validation and test set. Each one of the 5 bootstrap iterations is a new and unrelated experimental run. For each one of them a different training-test sets split was taken into account. Features were

selected analyzing the training set (the test set has never been taken into account for this part of the work). After the feature selection was executed according to both Shannon and Rényi min-entropy, we considered all the selected features adding one at each time. So, for each bootstrap iteration we had 50 steps, and in each step we added one of the selected features, we performed hyper-parameter tuning with 10 folds CV, we trained the model with the best parameters on the whole training set and we tested it on the test set (which the model had never seen so far). This procedure was performed both for SVM and ANN.

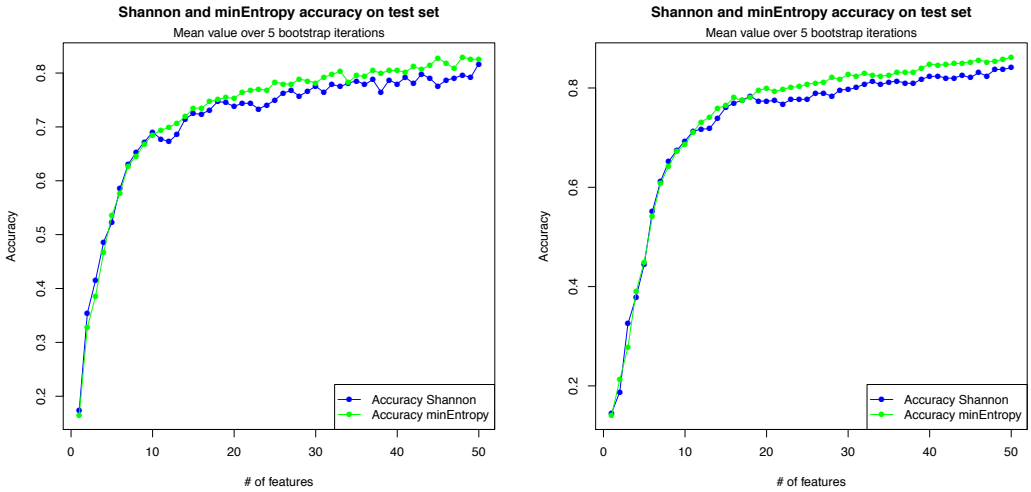


Figure 5.7 – Accuracy of the ANN and SVM classifiers on the Semeion dataset.

in all cases to the Semeion dataset, the features selected by the two metrics are not so different and we get similar results. Nonetheless the small differences reveal that with the Rényi min-entropy we achieve better results with the same number of features.

In the case of the Basehock dataset, where the nature and the amount of features are very different from those of the Semeion dataset, the advantage of the method using the Rényi entropy, with respect to the one which uses Shannon entropy, is even more pronounced.

We computed the average performances over the 5 iterations and the results are in fig. 5.5, fig. 5.6, and fig. 5.7. In all cases the feature selection method using Rényi min-entropy usually gave better results than Shannon, especially with the Basehock dataset.

5.4 Final remarks

Related work

Some of the works related to feature selection methods which are based on information theory have been already mentioned at the beginning of this chapter. In this section, we further comments on the those contributions and we cite new ones as well. For a more complete overview we refer to [BHS15], [VE14] and [BPZL12].

The approach most related to our proposal is that of [BPZL12] and [VE14]. We have already discussed and compared their method with ours earlier in this chapter. As to the work in [EK13], where the authors considered the Rényi entropies, it is important to notice that they used the notion of min-entropy defined in [Cac97]. This notion, as anticipated in section 2.2, has the unnatural characteristic that a feature may increase the entropy of the classification instead of decreasing it. It is clear, therefore, that basing a method on this notion of entropy could lead to strange results. Moreover, the results reported in [EK13] are relative to orders of Rényi entropies which are different from the min-entropy.

Another body of literature focuses on the two key concepts of *relevance* and *redundancy*. Relevance refers to the importance for the classification of the feature under consideration f^t, and it is in general modeled as $I_1(C; f^t)$. Redundancy represents how much the information of f^t is already covered by S. It is often modeled as $I_1(f^t, S)$. In general, we want to maximize relevance and minimize redundancy.

One of the first algorithms ever implemented was the MIFS algorithm proposed by [Bat94], based on a greedy strategy. At the first step it selects $f^1 = \operatorname{argmax}_{f_i \in F} I_1(C; f_i)$, and at step t it selects $f^t = \operatorname{argmax}_{f_i \in F \setminus S^{t-1}} [I_1(C, f_i) - \beta \sum_{f_s \in S^{t-1}} I_1(f_i, f_s)]$ where β is a parameter that controls the weight of the redundancy part. The mRMR approach (redundancy minimization and relevance maximization) proposed by [PLD05] is based on the same strategy as MIFS. However the redundancy term is now substituted by its mean over the elements of the subset S so to avoid its value to grow when new attributes are selected. In both cases, if relevance outgrows redundancy, it might happen that many features highly correlated and so highly redundant can still be selected. Moreover, a common issue with these two methods is that they do not take into account the conditional mutual information $I_1(C, f^t \mid S)$ for the choice of the next feature to be selected f^t.

More recent algorithms involve the ideas of joint mutual entropy $I_1(C; f_i, S)$ (JMI, [BHS15]) and conditional mutual entropy $I_1(C; f_i \mid S)$ (CMI, [Fle04]). The step for choosing the next feature with JMI is $f^t = \operatorname{argmax}_{f_i \in F \setminus S^{t-1}} \left\{ \min_{f_s \in S^{t-1}} I(C; f_i, f_s) \right\}$, while with CMI is $f^t = \operatorname{argmax}_{f_i \in F \setminus S^{t-1}} \left\{ \min_{f_s \in S^{t-1}} I(C; f_i \mid f_s) \right\}$. In both cases the already selected features are taken into account one by one when compared to the new feature f^t. The correlation between JMI and CMI is easy to prove [YM99]: $I_1(C; f_i, S) = H_1(C) - H_1(C \mid S) + H_1(C \mid S) - H_1(C \mid S) = I_1(C; S) + I(C; f_i \mid S)$.

Summary

We have proposed and formalized a method for feature selection based on a notion of conditional Rényi min-entropy. We have shown that the problem of selecting the optimal set of features with respect to the min-entropy, namely the S that satisfies (5.2), is NP-hard. Therefore, we have proposed an iterative greedy strategy of linear complexity to approximate the optimal subset of features with respect to the min-entropy. This strategy starts from the empty set, and adds a new feature at each step until we achieve the desired level of accuracy.

We have shown that our strategy is *locally optimal*, namely, at every step the new set of

features is the optimal one among those that can be obtained from the previous one by adding only one feature. (This does not imply, however, that the final result is *globally optimal*.) We think that leaves room for future implementations based on different notions of entropy which are the state-of-the-art in security and privacy, like the notion of g-vulnerability, which seems promising for its flexibility and capability to represent a large spectrum of possible classification strategies.

We have compared our approach with that based on Shannon entropy, and we have proven a negative result: neither of the two approaches is better than the other *in all cases*. We have used the Basehock, Semeion, and Gisette datasets and, although the two methods are, in general, incomparable, in the experiments we run, our method has always achieved better results.

CHAPTER 6
Conclusion

In this thesis, we have investigated how notions from information theory and ML can be utilized in the context of security and privacy, both in order to quantify the leakage of a given system and to build privacy protection mechanisms.

In the first presented contribution, we have proposed an approach to estimate the g-vulnerability of a system under the black-box assumption, using machine learning. We have introduced two pre-processing techniques to reduce the problem to that of learning the Bayes classifier on a set of pre-processed training data. Considering the applicability of our solution to real problems a matter of primary importance, we have proposed several experiments and we have compared our framework the frequentist approach, showing that the results are comparable when the observable domain is small, while our approach does much better on large observable domains. This is in line with what already observed in [CCP19] for the estimation of the Bayes error.

We have then moved on to study the problem of designing an obfuscation based privacy protection mechanism using an adversarial network paradigm inspired by the GAN paradigm. We have proposed a method based on adversarial nets to generate obfuscation mechanisms with a good trade-off between privacy and utility. The crucial feature of our approach is that the target function to minimize is the mutual information rather than the cross entropy. The applicability of our solution is an important aspect of our contribution. Therefore, we have shown applications to the case of location privacy, and experimented with a set of synthetic data and with data from Gowalla.

As to the third contribution, we have proposed a method for feature selection based on a notion of conditional Rényi min-entropy.

The line of research we have pursued in this work offers many opportunities for future work. For instance, as to the leakage estimation topic, we plan to thoroughly study how our framework adapts to specific real-life scenarios which have been subject of research in the last few years. For instance, we are interested in fingerprinting attacks [Che17, CHJ17] and the AES cryptographic algorithm [dCGRP19]. We also would like to consider the more general case, often considered in Information-flow security, of channels that have both "high" and "low" inputs, where the first are the secrets and the latter are data visible to, or even controlled by, the adversary.

A more ambitious goal would be to use our approach to minimize the g-vulnerability of complex systems, using a GAN based approach, along the lines of what has been discusses in chapter 4. Moreover, investigating ways to improve our current privacy protection mechanism based on the adversarial networks paradigm would be worthwhile. In general, using different loss functions would allow us to deploy mechanisms which are concerned with different notions of privacy.

A very important challenge is represented by the extension of the framework we presented to worst-case notions of privacy, such as differential privacy and geo-indistinguishability. To this end, it could be worth studying results obtained by applying a variant of differential privacy called Rényi differential privacy [Mir17], which is formulated in terms of divergence, and explore the applicability of the learning-based method for estimating f-divergences proposed in [RBD$^+$19].

A direct extension of the current implementation would be introducing a surrogate functions to deal with the estimation of the mutual information using losses which are more suitable for neural networks training in order to reduce the computational burden.

Moreover we plan to enhance the flexibility of the constraint on distortion (in the loss function), by requiring it to be *per user* rather than global. More specifically, we aim at producing obfuscation mechanisms that satisfy constraints stating that the expected displacement *for each user* is at most up to a certain threshold. The motivation is that different users may have different requirements. Another potential application is to encompass a notion of *fairness*, that can be obtained by requiring that the threshold is the same for everybody.

Bibliography

[AA16] Martín Abadi and David G. Andersen. Learning to protect communications with adversarial neural cryptography. *CoRR*, abs/1610.06918, 2016.

[AAC+15] Mário S. Alvim, Miguel E. Andrés, Konstantinos Chatzikokolakis, Pierpaolo Degano, and Catuscia Palamidessi. On the information leakage of differentially-private mechanisms. *J. of Comp. Security*, 23(4):427–469, 2015.

[AACP11] Mário S. Alvim, Miguel E. Andrés, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. On the relation between Differential Privacy and Quantitative Information Flow. In *Proc. of ICALP*, volume 6756 of *LNCS*, pages 60–76. Springer, 2011.

[ABCP13] Miguel E. Andrés, Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: differential privacy for location-based systems. In *Proc. of CCS*, pages 901–914. ACM, 2013.

[ABG+19] Ulrich Aïvodji, François Bidet, Sébastien Gambs, Rosin Claude Ngueveu, and Alain Tapp. Agnostic data debiasing through a local sanitizer learnt from an adversarial network approach. *CoRR*, abs/1906.07858, 2019.

[ACM+14] Mário S. Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. Additive and multiplicative notions of leakage, and their capacities. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 308–322. IEEE, 2014.

[ACM+16] Mário S. Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. Axioms for information leakage. In *Proceedings of the 29th IEEE Computer Security Foundations Symposium (CSF)*, pages 77–92, 2016.

[ACPS12] Mário S. Alvim, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Geoffrey Smith. Measuring information leakage using generalized gain functions. In Stephen Chong, editor, *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, pages 265–279. IEEE Computer Society, 2012.

[Ari75] Suguro Arimoto. Information measures and capacity of order α for discrete memoryless channels. In *Topics in Information Theory, Proc. Coll. Math. Soc. Janos Bolyai*, pages 41–52, 1975.

[Bat94] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. Neural Networks*, 5(4):537–550, 1994.

[BBR+18] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *Proceedings of the 35th Int. Conf. on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 531–540. PMLR, 2018.

[BCP14] Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Optimal geo-indistinguishable mechanisms for location privacy. In *Proc. of CCS*, 2014.

[BHS15] Mohamed Bennasar, Yulia Hicks, and Rossitza Setchi. Feature selection using joint mutual information maximisation. *Expert Syst. Appl*, 42(22):8520–8532, 2015.

[Bis07] Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007.

[BLM13] S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. OUP Oxford, 2013.

[Bor09] Michele Boreale. Quantifying information leakage in process calculi. *Inf. Comput*, 207(6):699–725, 2009.

[BPZL12] Gavin Brown, Adam Craig Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *JMLR*, 13:27–66, 2012.

[BS16] Nicolás E. Bordenabe and Geoffrey Smith. Correlated secrets in quantitative information flow. In *CSF*, pages 93–104. IEEE Computer Society, 2016.

[BWI16] Yuksel Ozan Basciftci, Ye Wang, and Prakash Ishwar. On privacy-utility tradeoffs for constrained data release mechanisms. In *Proc. of ITA*, pages 1–6. IEEE, 2016.

[CABP13] Konstantinos Chatzikokolakis, Miguel E. Andrés, Nicolás E. Bordenabe, and Catuscia Palamidessi. Broadening the scope of Differential Privacy using metrics. In *Proc. of PETS*, volume 7981 of *LNCS*, pages 82–102. Springer, 2013.

[Cac97] Christian Cachin. *Entropy Measures and Unconditional Security in Cryptography*. PhD thesis, ETH, 1997.

[CCG10] Konstantinos Chatzikokolakis, Tom Chothia, and Apratim Guha. Statistical measurement of information leakage. In *TACAS*, volume 6015 of *Lecture Notes in Computer Science*, pages 390–404. Springer, 2010.

[CCP19] Giovanni Cherubin, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. F-bleau: Fast black-box leakage estimation. *IEEE Symposium on Security and Privacy*, abs/1902.01350, 2019.

[CEP17] Kostantinos Chatzikokolakis, Ehab ElSalamouny, and Catuscia Palamidessi. Efficient utility improvement for location privacy. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2017(4):308–328, 2017.

[CFP19] Konstantinos Chatzikokolakis, Natasha Fernandes, and Catuscia Palamidessi. Comparing systems: max-case refinement orders and application to differential privacy. In *Proceedings of the 32nd IEEE Computer Security Foundations Symposium*, pages 442–457, Hoboken, United States, 2019.

[CG11] Tom Chothia and Apratim Guha. A statistical test for information leaks using continuous mutual information. In *CSF*, pages 177–190. IEEE Computer Society, 2011.

[Che17] Giovanni Cherubin. Bayes, not naïve: Security bounds on website fingerprinting defenses. *PoPETs*, 2017(4):215–231, 2017.

[CHJ17] Giovanni Cherubin, Jamie Hayes, and Marc Juárez. Website fingerprinting defenses at the application layer. *PoPETs*, 2017(2):186–203, 2017.

[CHM01] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantitative analysis of the leakage of confidential data. *Electr. Notes Theor. Comput. Sci*, 59(3):238–251, 2001.

[CKN13] Tom Chothia, Yusuke Kawamoto, and Chris Novakovic. A tool for estimating information leakage. In *International Conference on Computer Aided Verification (CAV)*, pages 690–695. Springer, 2013.

[CKN14] Tom Chothia, Yusuke Kawamoto, and Chris Novakovic. Leakwatch: Estimating information leakage from java programs. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS (2)*, volume 8713 of *Lecture Notes in Computer Science*, pages 219–236. Springer, 2014.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[CLWY18] Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70 – 79, 2018.

[CPP08a] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Inf. Comput*, 206(2-4):378–401, 2008.

[CPP08b] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. On the Bayes risk in information-hiding protocols. *Journal of Computer Security*, 16(5):531–571, 2008.

[Csi95] Imre Csiszár. Generalized cutoff rates and Rényi's information measures. *Trans. on Information Theory*, 41(1):26–34, 1995.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. J. Wiley & Sons, Inc., 1991.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. J. Wiley & Sons, Inc., second edition, 2006.

[CY16] Paul Cuff and Lanqing Yu. Differential privacy as a mutual information constraint. In *Proc. of CCS*, CCS '16, pages 43–54. ACM, 2016.

[dCGRP19] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best Information is Most Successful - Mutual Information and Success Rate in Side-Channel Analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):49–79, 2019.

[De12] Anindya De. Lower bounds in differential privacy. In *Proc. of TCC*, pages 321–338. Springer, 2012.

[DG17] Dheeru Dua and Casey Graff. UCI Machine Learning Repository (Heart Disease Data Set), 2017.

[DGL96] Luc Devroye, László Györfi, and Gábor Lugosi. *Vapnik-Chervonenkis Theory*, pages 187–213. Springer New York, New York, NY, 1996.

[DJW13] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local privacy and statistical minimax rates. In *Proc. of FOCS*, pages 429–438. IEEE Computer Society, 2013.

[DMNS06] Cynthia Dwork, Frank Mcsherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *In Proceedings of the Third Theory of Cryptography Conference (TCC)*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[DR94] Scott Davies and Stuart Russell. Np-completeness of searches for smallest possible feature sets, 1994.

[Dwo06] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *33rd International Colloquium on Automata, Languages and Programming (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[EK13] Tomomi Endo and Mineichi Kudo. Weighted Naïve Bayes Classifiers by Renyi Entropy. In *Proc. of CIARP*, volume 8258 of *LNCS*, pages 149–156. Springer, 2013.

[ES16] Harrison Edwards and Amos J. Storkey. Censoring representations with an adversary. In *Proc. of ICLR*, 2016.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proc. of CCS*, CCS '15, pages 1322–1333. ACM, 2015.

[Fle04] François Fleuret. Fast binary feature selection with conditional mutual information. *JMLR*, 5:1531–1555, 2004.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth Int. Conf. on AI and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. PMLR, 2010.

[GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *JMLR*, 3:1157–1182, 2003.

[GGH12] Sébastien Gambs, Ahmed Gmati, and Michel Hurfin. Reconstruction attack through classifier analysis. *DBSEC - 26th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy - 2012*, July 11 2012.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[Ham17] Jihun Hamm. Minimax filter: Learning to preserve privacy from inference attacks. *J. Mach. Learn. Res.*, 18(1):4704–4734, 2017.

[HKC+17] Chong Huang, Peter Kairouz, Xiao Chen, Lalitha Sankar, and Ram Rajagopal. Context-aware generative adversarial privacy. *Entropy*, 19(12), 2017.

[HMDC19] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: membership inference attacks against generative models. *PoPETs*, 2019(1):133–152, 2019.

[HO18] Jamie Hayes and Olga Ohrimenko. Contamination attacks and mitigation in multi-party machine learning. In *Proc. of NIPS*, pages 6604–6616. Curran Associates Inc., 2018.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, 2001.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009.

[JDM00] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[JG18] Jinyuan Jia and Neil Zhenqiang Gong. Attriguard: A practical defense against attribute inference attacks via adversarial machine learning. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 513–529. USENIX Association, 2018.

[Kar72] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, 1972.

[KB07] Boris Köpf and David A. Basin. An information-theoretic model for adaptive side-channel attacks. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 286–296. ACM, 2007.

[KD09] Boris Köpf and Markus Dürmuth. A provably secure and efficient countermeasure against timing attacks. In *Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*, CSF '09, pages 324–335, USA, 2009. IEEE Computer Society.

[KM17] M. H. R. Khouzani and Pasquale Malacaria. Leakage-minimal design: Universality, limitations, and applications. In *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pages 305–317. IEEE Computer Society, 2017.

[KM18] M. H. R. Khouzani and Pasquale Malacaria. Optimal channel design: A game theoretical analysis. *Entropy*, 20(9):675, 2018.

[KMN+16] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016.

[LK] Jure Leskovec and Andrej Krevl. The Gowalla dataset (Part of the SNAP collection). https://snap.stanford.edu/data/loc-gowalla.html. [Accessed 18 May 2019].

[LLWW18] Jinghua Liu, Yaojin Lin, Shunxiang Wu, and Chenxi Wang. Online multi-label group feature selection. *Knowledge-Based Systems*, 143:42 – 57, 2018.

[LY05] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowledge and Data Engineering*, 17(4):491–502, 2005.

[Mir13] Darakhshan J. Mir. Information-theoretic foundations of differential privacy. In *Proc. of FPS*, pages 374–381. Springer Berlin Heidelberg, 2013.

[Mir17] Ilya Mironov. Rényi differential privacy. In *Proc. of CSF*, pages 263–275, 2017.

[MMM10] Annabelle McIver, Larissa Meinicke, and Carroll Morgan. Compositional Closure for Bayes Risk in Probabilistic Noninterference. In *Proc. of ICALP*, volume 6199 of *LNCS*, pages 223–235. Springer, 2010.

[Moo65] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117, 1965.

[MSCS19] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *Proc. of S&P*, pages 497–512. IEEE, 2019.

[Nak18] Songyot Nakariyakul. High-dimensional hybrid feature selection using interaction information-guided search. *Knowledge-Based Systems*, 145:59 – 66, 2018.

[NCT16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Proc. of NIPS*, pages 271–279, 2016.

[NS08] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125. IEEE Computer Society, 2008.

[OTPG17] Simon Oya, Carmela Troncoso, and Fernando Pérez-González. Back to the drawing board: Revisiting the design of optimal location privacy-preserving mechanisms. In *Proc. of CCS*, CCS '17, pages 1959–1972. ACM, 2017.

[Ped11] F. et al. Pedregosa. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011.

[PLD05] Hanchuan Peng, Fuhui Long, and Chris H. Q. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell*, 27(8):1226–1238, 2005.

[PR18] Catuscia Palamidessi and Marco Romanelli. Feature selection with rényi min-entropy. In Luca Pancioni, Friedhelm Schwenker, and Edmondo Trentin, editors, *Artificial Neural Networks in Pattern Recognition - 8th IAPR TC3 Workshop, ANNPR 2018, Siena, Italy, September 19-21, 2018, Proceedings*, volume 11081 of *Lecture Notes in Computer Science*, pages 226–239. Springer, 2018.

[PSB+18] Emmanuel Prouff, Rémi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ascad database. *IACR Cryptology ePrint Archive*, 2018:53, 2018.

[PTC18] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. In *Proc. of NDSS*. The Internet Society, 2018.

[RBD+19] Paul K. Rubenstein, Olivier Bousquet, Josip Djolonga, Carlos Riquelme, and Ilya O. Tolstikhin. Practical and consistent estimation of f-divergences. In *Proc. of NIPS*, pages 4072–4082, 2019.

[RCP20] Marco Romanelli, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Generating optimal privacy-protection mechanisms via machine learning. In *Proceedings of the IEEE International Symposium on Computer Security Foundations (CSF)*, 2020.

[RCPP20] Marco Romanelli, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Pablo Piantanida. Estimating g-leakage via machine learning, 2020.

[Rén61] Alfréd Rényi. On Measures of Entropy and Information. In *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics, and Probability*, pages 547–561, 1961.

[RLR18] Zhongzheng Ren, Yong Jae Lee, and Michael S. Ryoo. Learning to anonymize faces for privacy preserving action detection. In *Proc. of ECCV*, volume 11205 of *LNCS*, pages 639–655. Springer, 2018.

[Sho15] Reza Shokri. Privacy games: Optimal user-centric data obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2015(2):299–315, 2015.

[Sib69] R. Sibson. Information radius. *Z. Wahrscheinlichkeitsth. und Verw. Geb*, 14:149–161, 1969.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan Prescott Adams. Practical bayesian optimization of machine learning algorithms. In *Proc. of NIPS 2012*, pages 2960–2968, 2012.

[Smi09] Geoffrey Smith. On the foundations of quantitative information flow. In Luca de Alfaro, editor, *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5504 of *Lecture Notes in Computer Science*, pages 288–302. Springer, 2009.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning : from theory to algorithms*. Cambridge University Press, 2014.

[SSGC17] Razieh Sheikhpour, Mehdi Agha Sarram, Sajjad Gharaghani, and Mohammad Ali Zare Chahooki. A survey on semi-supervised feature selection methods. *Pattern Recognition*, 64:141 – 158, 2017.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proc. of S&P*, pages 3–18. IEEE Computer Society, 2017.

[STBH11] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. Quantifying location privacy. In *IEEE Symposium on Security and Privacy*, pages 247–262. IEEE Computer Society, 2011.

[Sto77] C. J. Stone. Consistent nonparametric regression. *Annals of Statistics*, 5(4):595–645, 1977.

[STT+12a] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Protecting location privacy: optimal strategy against localization attacks. In *Proc. of CCS*, pages 617–627. ACM, 2012.

[STT+12b] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Protecting location privacy: optimal strategy against localization attacks. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 617–627. ACM, 2012.

[STT17] Reza Shokri, George Theodorakopoulos, and Carmela Troncoso. Privacy games along location traces: A game-theoretic framework for optimizing location privacy. *ACM Trans. on Privacy and Security*, 19(4):11:1–11:31, 2017.

[SV06] Nandakishore Santhi and Alexander Vardy. On an improvement over Rényi's equivocation bound, 2006. Presented at the 44-th Annual Allerton Conf. on Communication, Control, and Computing, September 2006. Available at http://arxiv.org/abs/cs/0608087.

[TWI17] Ardhendu Tripathy, Ye Wang, and Prakash Ishwar. Privacy-preserving adversarial networks. *CoRR*, abs/1712.07008, 2017.

[Val84] Leslie G. Valiant. A theory of the learnable. In Richard A. DeMillo, editor, *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 436–445. ACM, 1984.

[VE14] Jorge R. Vergara and Pablo A. Estévez. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24(1):175–186, 2014.

[Wol96] David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[YM99] Howard Hua Yang and John Moody. Feature selection based on joint mutual information. In *In Proceedings of Int. ICSC Symposium on Advances in Intelligent Data Analysis*, pages 22–25, 1999.

[ZB05] Ye Zhu and Riccardo Bettati. Anonymity vs. information leakage in anonymity systems. In *Proc. of ICDCS*, pages 514–524. IEEE, 2005.

DOCTORAL SCHOOL

Titre : Méthodes d'apprentissage machine pour la protection de la vie privée: mesure de leakage et design des mécanismes

Mots clés : Estimation des fuites, protection de la vie privée, apprentissage machine

Résumé : Dans cette thèse de doctorat, nous étudions comment les notions de théorie de l'information et de ML peuvent être utilisées pour mieux mesurer et comprendre les informations divulguées par les données et/ou les modèles, et pour concevoir des solutions visant à protéger la confidentialité des informations partagées. Nous explorons l'application du ML pour estimer la fuite d'informations d'un système dans le scénario black-box où les seules informations disponibles sont des paires input-output. Les travaux précédents se sont concentrés sur l'estimation des fréquences conditionnelles d'entrée-sortie et, plus récemment, l'estimation de l'erreur de Bayes a été étudiée à l'aide de modèles ML et s'est avérée plus précise. Nous proposons une nouvelle approche basée sur la ML, qui repose sur le pre-processing des données, pour effectuer une estimation de la g-vulnerability, une mesure plus souple de la fuite, qui englobe l'attaquant Bayes et plusieurs autres types d'adversaires. Nous étudions formellement la capacité d'apprentissage pour toutes les distributions de données et évaluons les performances dans divers cadres expérimentaux. Ensuite, nous nous attaquons au problème de l'obscurcissement des informations sensibles tout en préservant leur utilité, et nous proposons une approche ML inspirée du paradigme GAN pour produire un générateur d'obscurcissement optimal pour protéger les données, alors que classificateur essaie de désobstruer les données. Nous appliquons notre méthode au cas de la protection de la vie privée en matière de localisation. La performance du mécanisme d'obfuscation obtenu est évaluée en fonction de l'erreur de Bayes, qui représente l'adversaire le plus puissant possible. Enfin, nous considérons que, compte tenu des valeurs des classes (secrètes) et des caractéristiques (observables) dans les problèmes de classification, la mesure de la fuite d'un tel système est une stratégie visant à distinguer les caractéristiques les plus et les moins informatives. Nous comparons l'information mutuelle basée sur l'entropie de Shannon à celle basée sur la min-entropie de R'enyi, tant du point de vue théorique qu'expérimental: selon l'ensemble de données considéré, parfois la méthode basée sur l'entropie de Rényi est plus performante que l'autre, et parfois le contraire se produit.

Title : Machine Learning methods for privacy protection: leakage measurement and mechanism design

Keywords : Leakage estimation, privacy protection, machine learning

Abstract : In this PhD thesis, we study how notions of information theory and ML can be used to better measure and understand the information leaked by data and / or models, and to design solutions to protect the privacy of the shared information. We first explore the application of ML to estimate the information leakage of a system in the the black-box scenario where the only available information are pairs of input-output data samples. Previous works focused on estimate the input-output conditional frequencies, and, more recently, the estimation of the Bayes error was investigated using ML models and it has been shown to be more accurate. We propose a novel ML based approach, that relies on data preprocessing, to perform black-box estimation of the g-vulnerability a more flexible measure of leakage, which encompasses the Bayes attacker and several other types of adversaries. We formally study the learnability for all data distributions and evaluate performances in various experimental settings. Secondly, we address the problem of obfuscating sensitive information while preserving utility, and we propose a ML approach inspired by the GAN paradigm. The generator net tries to produce an optimal obfuscation mechanism to protect the data, and the classifier tries to de-obfuscate the data. By letting the two nets compete against each other, the mechanism improves its degree of protection, until an equilibrium is reached. We apply our method to the case of location privacy. The performance of the obtained obfuscation mechanism is evaluated in terms of the Bayes error, which represents the strongest possible adversary. Finally, we consider that, given classes (secrets) and features' values (observables) in classification problems, measuring the leakage of a system is a strategy to tell the most and least informative features apart. The prediction power stems from the correlation, i.e., the mutual information, between features and labels. We compare the Shannon entropy based mutual information to the Rényi min-entropy based one, both from the theoretical and experimental point of view showing that, depending on the considered dataset, sometimes the Shannon entropy based method outperforms the Rényi min-entropy based one and sometimes the opposite occurs.

Institut Polytechnique de Paris
91120 Palaiseau, France