McGill University

On Privacy for Reinforcement Learning Algorithms

by

Maziar Gomrokchi

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Science
School of Computer Science, McGill University, Montreal

December 2021

McGill University, Montreal

Abstract

Faculty of Science
School of Computer Science, McGill University, Montreal

Doctor of Philosophy

by Maziar Gomrokchi

Privacy has become one of the fundamental concerns in the industrial deployment of machine learning algorithms. While designing adversarial attacks reveals the potential level of privacy leakage in these algorithms, privacy-preserving algorithm design techniques propose solutions to protect individuals' privacy at certain levels. Despite the recent advancements in the design and development of privacy-preserving machine learning algorithms, research on the subject of private reinforcement learning is still in its infancy. In this thesis, we develop the first differentially private reinforcement learning algorithms for the problem of evaluating a given way of behaving (policy) and provide a comprehensive analysis of the privacy-accuracy trade-off in the proposed algorithms. We subsequently introduce a generic sub-sampling framework to improve the utility of these private algorithms. Theoretically, we show that under certain assumptions, our proposed sub-sampling framework significantly improves the utility of the underlying private algorithms. Empirically, in a full Markov Decision Process setting, we observe that the sub-sampling framework amplifies the accuracy of the proposed differentially private algorithms across different privacy regimes. Finally, we establish the first membership inference attack framework against a state-of-the-art deep reinforcement learning model. We demonstrate the vulnerability of deep reinforcement learning to membership inference attacks in collective (*i.e.* consisting of a collection of data points) and individual membership inference modes. We show that the learning stage of the deep reinforcement learning agent (how close the learned policy is to the optimal policy) and the temporal correlation among transition tuples within input trajectories have significant impact on the membership inference accuracy.

Université McGill

Abrégé

Faculté des Sciences
School of Computer Science, Université McGill, Montréal

Doctor of Philosophy

par Maziar Gomrokchi

La confidentialité est devenue l'une des préoccupations fondamentales dans le déploiement industriel des algorithmes d'apprentissage machine. Alors que la conception d'attaques adversariales révèle le niveau potentiel de perte de la vie privée de ces algorithmes, les techniques de conception d'algorithmes qui ont pour bût de préserver la vie privée, proposent des solutions pour protéger la vie privée des individus à certains niveaux. Malgré les récents progrès de la conception et le développement d'algorithmes d'apprentissage machine préservant la confidentialité, la recherche sur le sujet de l'apprentissage par renforcement privé en est encore dans son enfance. Tout au long de cette thèse, nous développons les premiers algorithmes différentiels d'apprentissage par renforcement privés et fournissons une analyse complète du compromis confidentialité-précision dans les algorithmes proposés.Nous introduisons par la suite un cadre de sous-échantillonnage générique pour améliorer l'utilité des algorithmes privés. Théoriquement, nous montrons que sous certaines hypothèses, notre cadre de sous-échantillonnage proposé améliore considérablement l'utilité des algorithmes privés. Empiriquement, dans un cadre de processus décisionnel de Markov complet, nous observons que le cadre de sous-échantillonnage amplifie la précision des algorithmes différentiellement privés proposés dans différents régimes de confidentialité. Enfin, nous établissons le premier cadre d'attaque par inférence d'appartenance contre un modèle d'apprentissage par renforcement profond à la pointe de la technologie. Nous démontrons la vulnérabilité de l'apprentissage par renforcement profond aux attaques par inférence d'appartenance dans les modes d'inférence d'appartenance collective (*i.e.* consistant en une collection de points de données) et individuelle. Nous montrons que l'état d'apprentissage de l'agent d'apprentissage par renforcement profond (à quel point la politique apprise est proche de la politique optimale) et la corrélation temporelle entre les tuples de transition avec les trajectoires d'entrée ont un impact significatif sur la précision de l'inférence des membres.

Contribution to Original Knowledge

This thesis contributes to the understanding of privacy-preserving reinforcement learning. More specifically, it addresses the problem of private policy evaluation and membership inference attacks in reinforcement learning by:

1. introducing privacy-preserving policy evaluation algorithms in reinforcement learning, including:
    - two novel privacy-preserving algorithms for Monte Carlo policy evaluation in the Markov Decision Process setting.
    - a comprehensive theoretical analysis of privacy-utility trade-off in the proposed algorithms.
    - empirical analysis of the proposed algorithms in the full MDP setting.
2. designing a novel sub-sampling framework for utility amplification of privacy-preserving algorithms through:
    - defining novel sensitivity and utility measures tailored to the proposed sub-sampling framework,
    - a comprehensive comparison between the theoretical utility bounds associated with the baseline private algorithms and the ones obtained in the proposed sub-sampling framework,
    - empirical evaluation of the sub-sampling impact on the utility of the baseline private algorithms.
3. proposing a novel framework to investigate membership inference attacks against deep reinforcement learning models, including:
    - a unified framework for attacks against deep reinforcement learning models,
    - two novel deep attack classifiers designed to perform inference attacks in individual and collective modes,
    - empirical evaluation of the proposed framework against a state-of-the-art deep reinforcement learning model in high-dimensional continuous control environments.

Contribution of Authors

- Chapter 3 presents the first privacy-preserving policy evaluation algorithm that appeared in Balle et al. (2016) that I second-authored. During this work I initiated the ideas and carried out the empirical evaluation. Borja Balle carried out the theoretical analysis with my collaboration. My advisor, Doina Precup, helped with the writing and advised on the research.
- Chapter 4 introduces a novel utility amplification framework for privacy-preserving algorithms, which is empirically benchmarked as a wrapper on the algorithms proposed in Chapter 3. I have been the main contributor to this work. My collaborator Borja Balle helped me with the formalism of the framework and my advisor, Doina Precup, helped with the writing and advised on the research. The paper, on which I am first author, is ready for submission to an upcoming machine learning venue.
- Chapter 5 proposes the first membership inference attack framework against deep reinforcement learning models. I was the main initiator of the ideas, designer of the framework and responsible for the writing. Susan Amin and Hossein Aboutalebi helped my in both writing and experiments of this project and Doina Precup and Alex Wong helped in the writing and advised on the research direction of this project. A paper on this work, on which I am the first author, was published at NeurIPS 2020 Privacy-Preserving Machine Learning (PriML and PPML Joint Edition) workshop. The work was subsequently extended and is currently under review in an upcoming privacy and security venue.

0.1 Other Individual Contributions

During my Ph.D., apart from the research that I have conducted on the privacy of reinforcement learning algorithms, I was actively involved in the following side projects:

- *Reproducibility of benchmarked deep reinforcement learning tasks for continuous control* (Islam et al., 2017), which I third-authored. (Published in Workshop of Reproducibility in Machine Learning, ICML 2017)
- *Locally Persistent Exploration in Continuous Control Tasks with Sparse Rewards* (Amin et al., 2020), which I jointly first-authored. (Accepted in International Conference on Machine Learning (ICML), 2021)
- A Survey of Exploration Methods in Reinforcement Learning (Amin et al., 2021), which I second-author. (submitted at Journal of Artificial Intelligence Research (JAIR))

Acknowledgements

First and foremost, I would like to thank my wife and collaborator, Susan Amin, for her constant optimism and support throughout my Ph.D. work.

I would also like to express my great appreciation to my supervisor, Doina Precup for her advice, expertise, and patience, which are the reason I have had the opportunity to conduct my research in such fascinating topics. I sincerely thank my bothers, my sister and my parents for supporting me spiritually and mentally throughout my PhD and my life in general.

Finally, I wish to express my deep thanks to my collaborators: Susan Amin, Borja Balle, Hossein Aboutalebi, Harsh Satija, Herke van Hoof, Guillaume Rabusseau, Guillaume Lam, and Riashat Islam for their priceless help with the projects.

Contents

Abstract **i**

Abrégé **ii**

Contribution to Original Knowledge **iii**

Contribution of Authors **iv**

0.1 Other Individual Contributions v

Acknowledgements **vi**

1 Introduction and motivation **1**

1.1 Overview and Contributions 3

2 Background and related work **5**

2.1 Reinforcement Learning 5

2.1.1 Policy Evaluation 8

2.2 Batch Off-policy Deep Reinforcement Learning 10

2.3 Differential privacy 13

2.3.1 Output perturbation mechanism 14

2.3.2 Private Reinforcement Learning 15

2.4 Membership Inference Attacks 16

2.4.1 Membership inference in supervised machine learning. 17

2.4.2 Membership inference attack in reinforcement learning 18

2.4.3 Performance Metrics 19

3 Private Monte Carlo Policy Evaluation **22**

3.1 Introduction and motivation 22

3.2 Background and notation 23

3.2.1 Monte Carlo Policy Evaluation 24

3.3 Private First-Visit Monte Carlo Algorithms . . . 25
3.3.1 Algorithm DP-LSW . . . 25
3.3.2 Algorithm DP-LSL . . . 27
3.4 Privacy Analysis . . . 28
3.4.1 Smoothed Gaussian Perturbation . . . 29
3.4.2 Privacy Analysis of DP-LSW . . . 34
3.4.3 Privacy Analysis of DP-LSL . . . 38
3.5 Utility Analysis . . . 41
3.6 Utility Analysis of DP-LSW . . . 42
3.7 Utility Analysis of DP-LSL . . . 45
3.8 Experiments . . . 49
3.9 Discussion . . . 50

4 Utility Amplification via Sub-Sampling 53
4.1 Review of Sub-Sampling Techniques Used in Differential Privacy . . . 54
4.2 Notation and Framework . . . 55
4.3 Average Utility Case Study: Gaussian Output Perturbation Mechanism . 57
4.4 Privacy of Sub-Sample-and-Average Gaussian Mechanism . . . 59
4.5 Average Sub-sampling Sensitivity of Linear Queries . . . 61
4.6 Average Utility of Sub-sample and Average Gaussian Mechanism: Average Case . . . 62
4.7 Average Utility of Sub-sample and Average Gaussian Mechanism: Extreme Case . . . 68
4.8 Algorithm and Experimental Evaluation . . . 69
4.8.1 Experiment Setup . . . 69
4.9 Discussion . . . 72

5 Membership Inference Attacks Against Deep Reinforcement Learning 73
5.1 Membership Inference Attack in Machine Learning . . . 74
5.2 Membership Inference Attack in Deep Reinforcement Learning . . . 75
5.3 Problem Statement . . . 77
5.4 Related Work . . . 78
5.5 Attack Framework . . . 80
5.6 Experimental Setup . . . 82
5.6.1 Environments and RL Setting . . . 83
5.6.2 Data Augmentation . . . 84
5.6.3 Attack Classifier Architecture . . . 84
5.6.3.1 Individual Mode Attack Classifier Architecture . . . 86
5.6.3.2 Collective Mode Attack Classifier Architecture . . . 86
5.7 Results and Discussion . . . 87
5.7.1 Collective vs. Individual MIAs . . . 89
5.7.2 The Impact of $T_{\max}$ . . . 90
5.7.3 Clipping Length Impact . . . 90
5.7.4 Temporal Correlation . . . 92

5.8 Discussion 93

6 Conclusions and Future Work **95**

6.1 Main Findings 95

6.2 Future directions 96

6.2.1 Algorithm Design Extensions 97

6.2.2 Membership Inference Attack Extensions 97

Bibliography **99**

To my wife Susan, my parents and my sister and brothers.

Chapter 1

Introduction and motivation

Machine learning (ML) methods have a distinctive potential to revolutionize many domains, such as healthcare, robotics, transportation and natural language processing. Over the past decade, we have witnessed remarkable success in the application of traditional ML (Bishop, 2006) and deep learning (Goodfellow et al., 2016) methods in many domains, including those in which data can be sensitive in nature, such as medical domains (McKinney et al., 2020). The rise of data-driven solutions and the explosion of data size due to the recent progress in high-performance computing systems have equipped ML tools with a great source of information, which has led to significantly improved modelling and prediction accuracy. While these improvements are notable, it is also apparent that many existing ML applications require access to individuals' private and personal information to achieve the desired level of accuracy. Broad applicability and vast deployment of ML methods in industrial-scale problems pose privacy challenges to the field. Hence, an increasing amount of research and development effort in ML is devoted to the development of ML-based systems that, while delivering state-of-the-art performance, also preserve the privacy of the underlying data.

Despite the numerous studies and efforts in privacy preservation, we still witness reports on privacy breaches in a variety of domains (Narayanan and Shmatikov, 2006; Douriez et al., 2016; Pandurangan, 2014; Yeom et al., 2018). In fact, it has been shown that data anonymization and other similar techniques are not sufficient to prevent tracking personal information, and that machine learning can be used to reverse-engineer such information (Rocher et al., 2019). Thus, the general public has become aware of the potential danger of using non-private ML services (Cormode, 2011; Fredrikson et al., 2014; Li et al., 2013; Shokri et al., 2017; Brickell and Shmatikov, 2008). Due to this potential harm and lack of practical privacy-preserving ML solutions and algorithms, many industries have adopted a conservative approach in data sharing. Since 2006, when Dwork (2006) proposed *Differential Privacy* as a mathematically rigorous privacy standard, there has been an extensive effort to develop privacy-preserving ML algorithms (Friedman and Schuster, 2010; Dwork et al., 2014, 2006; Dwork, 2008; Ji et al., 2014; Arachchige et al., 2019; Mohammed et al., 2013; Alhadidi et al., 2012). On the other hand, there have been attempts to design adversarial privacy attacks, in order to identify and expose the vulnerabilities of existing ML solutions against such attacks (Shokri et al., 2017; Sablayrolles et al., 2019; Hu et al., 2021; Su et al., 2020).

However, most of the existing research effort has been devoted to privacy in supervised learning, where the data is assumed to be labelled and to be governed by i.i.d. assumptions. This makes both theoretical analysis and algorithm development easier, as one can consider, for example, introducing various forms of noise in individual examples in a dataset. This allows achieving reasonable trade-offs between privacy requirements and the resulting accuracy, or utility, of models trained on private data. We are instead interested in the setting of reinforcement learning (RL), in which an agent interacts with its environment over a period of time, taking actions and receiving observations and rewards. RL brings some significant complications compared to supervised learning, both for theoretical analysis and algorithm design. First, in sequential tasks, the inputs are not i.i.d., but instead depend on each other. Moreover, the data distribution from which the agent samples is not fixed, and instead depends on its action choices. These choices typically vary over time, as the goal of an RL agent is to find a way of behaving (also called a policy) which maximizes the expectation of its long-term cumulative reward (more details on the specifics of these notions are given in Chapter 2). Yet RL algorithms are increasingly making their way into practical applications with sensitive data, from medicine to finance (Zhou et al., 2021; Maeda et al., 2020). Hence, it is imperative to study their privacy properties and to understand how vulnerable they are to privacy attacks.

In recent years, there have been some works on privacy-preserving RL algorithms, e.g. Balle et al. (2016); Wang and Hegde (2019); Suriyakumar et al. (2020); Vietri et al. (2020). In this thesis, we aim to further contribute to the study of privacy in RL from both the algorithm designer and the adversarial point of view. Thus, we address three main questions:

1. Can we develop privacy-preserving policy evaluation algorithms?

2. Can we amplify the utility of private policy evaluation algorithms using well-established techniques such as sub-sampling?

3. How vulnerable is reinforcement learning to membership inference attacks?

The first question is motivated by the fact that policy evaluation, i.e. computing the expected long-term return of a given policy, is a core component of many RL algorithms, including algorithms that try to find an optimal way of behaving. Such algorithms often evaluate the existing policy first, and then improve it by increasing the probability of taking actions that have higher value, as established in this estimation process. Hence, understanding privacy for this problem, and developing private algorithms for policy evaluation is an important step towards privacy-preserving RL in general. Policy evaluation already presents the difficulty of the agent observing a non-iid stream of data, as both the inputs and the outputs observed on a trajectory are correlated. However, the input distribution observed by the agent is fixed, as the policy does not change, which makes this problem more tractable, intuitively, than the full control case, when the policy changes over time.

The second question is motivated by the fact that the utility loss of privacy-preserving policy evaluation can be high, i.e. the error in estimating returns can be quite big. This would limit the practical use of such algorithms. Hence, we study the use of well-established sub-sampling techniques to improve the utility, while is preserving privacy.

The final question is motivated by a recent body of work which shows that various privacy-preserving ML algorithms are still vulnerable to a particular kind of attack, called membership inference attack, in which someone tries to identify if a particular data point has been used in training a particular ML model. This motivated us to understand if RL algorithms, specifically deep RL, could be vulnerable to this kind of attack.

1.1 Overview and Contributions

The thesis is structured as follows. Chapter 2 reviews the necessary background on reinforcement learning, differential privacy and membership inference attack design in machine learning. In Chapter 3, we answer the first research question by developing privacy-preserving policy evaluation algorithms and empirically evaluating their performance in a synthetic, simulated domain. We carry out a comprehensive theoretical and empirical analysis of the privacy-utility trade-off of the proposed algorithms. In Chapter 4, we first provide a set of tools and background required for a systematic approach towards amplifying the utility of a class of algorithms known as linear queries. We subsequently employ these tools and propose a subroutine that amplifies the utility of the algorithms introduced in Chapter 3. We show empirically the substantial positive impact of the proposed approach on the utility of these algorithms. In Chapter 5, we tackle the third question by designing a generic membership inference attack framework tailored to non-private RL algorithms. We analyze empirically the vulnerability of a state-of-the-art deep RL algorithm to membership inference attacks. Finally, we conclude in Chapter 6 with a summary of the contributions and results in this dissertation, and discuss possible future work directions. Please refer to the preamble of the thesis for a detailed discussion of contributions as well.

Chapter 2

Background and related work

This chapter presents background on differential privacy and reinforcement learning that is required in order to design privacy-preserving policy evaluation algorithms and membership inference attacks against reinforcement learning models. We begin with an overview of the general RL paradigm in Section 2.1, followed by Differential Privacy background in Section 2.3. We then provide some examples of prior work on privacy-preserving algorithms in reinforcement learning, and finally, in Section 2.4, we discuss the prior work on membership inference attacks in machine learning more broadly, and then specifically in reinforcement learning. Further, more detailed background is also included in later chapters as needed.

2.1 Reinforcement Learning

Reinforcement learning was inspired by behavioral psychology and theories of animal learning but its modern version developed at the confluence of control theory, operations research and machine learning (see Sutton and Barto (1998) for a comprehensive introduction). In reinforcement learning, an agent interacts with its environment sequentially, receiving observations and taking actions. The agent's actions influence the observations received. Additionally, the agent receives a numerical signal called reward. The agent's goal is to take actions in such a way as to maximize the expected value of a cumulative function of the rewards received (with several such functions considered in different works).

The agent's task is usually formalized as a Markov Decision Process (MDP). A fixed-horizon, discrete-time MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, p_0, T \rangle$ consisting of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, and initial state distribution p_0, a transition probability distribution $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathbb{P}(\mathcal{S})$, which assigns to each state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ a probability measure over states in $\mathcal{S}$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and the number of time steps in an episode T. We note that the rewards are in fact typically assumed to be bounded in absolute value by $R_{\max}$. We also note that rewards can depend in general on the triple s_t, a_t, s_{t+1}, but we do not consider this case for ease of development; there are straightforward procedures for extending results to this case.

At each time-step $t = 0, 1, 2, \ldots$, the agent's state is $s_t \in \mathcal{S}$ and it chooses an action $a_t \in \mathcal{A}$. The next state of the agent, s_{t+1}, is determined stochastically by $\mathcal{P}(s_{t+1}|s_t, a_t)$ and the reward obtained by the agent on this transition, $r_t = r(s_t, a_t)$, is computed by the reward function. We note that the initial state s_0 is sampled from p_0, and that in the fixed horizon setting, the agent's episode (i.e. interaction with the environment) terminates after T transitions, in state s_T. A *trajectory* is a sequence of states, actions and rewards obtained by the agent. The *total return* of the agent on a trajectory in a fixed-horizon MDP, starting from time step $0 \leq t < T$, is given by the sum of rewards:

$$G_t = \sum_{k=0}^{T-t-1} r_{t+k} \tag{2.1}$$

If the agent's interaction is assumed to continue for an unknown amount of time, or possibly forever, the task can be modelled as a discounted discrete-time MDP, modelled as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, p_0, \gamma \rangle$, where all elements are as described above and $\gamma \in [0, 1)$ is the discount factor, which is used to incentivize the agent to pay less attention to rewards that occur in the far future. The process of interaction between the agent and the environment is as described above, except a trajectory might continue forever, or might stop only upon entering a designated terminal state. The *discounted return* on a given trajectory, from time t onward, is calculated as follows:

$$G_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{2.2}$$

A *Markovian stochastic policy*, $\pi : \mathcal{S} \to \mathbb{P}(\mathcal{A})$, maps states to a probability distribution over actions. Given a state s, the policy provides a conditional distribution over actions, conditioned on s, $\pi(a|s)$. A *deterministic Markovian policy* $\pi : \mathcal{S} \to \mathcal{A}$ is a map which

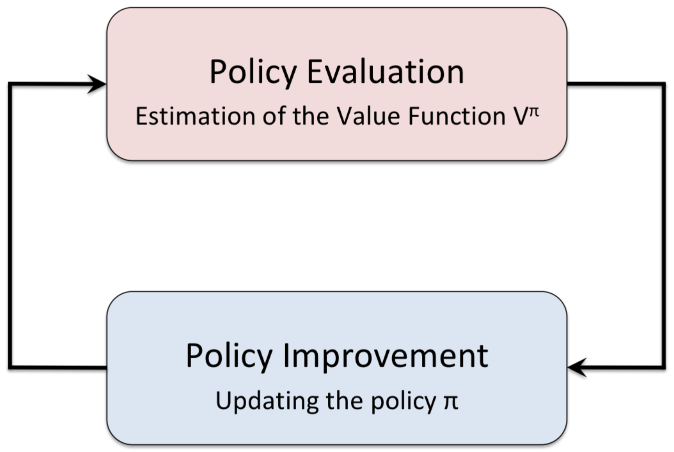


FIGURE 2.1: A schematic of the *policy iteration* method.

assigns an action to each state. In the stochastic policy setting, the agent samples from the policy to select the action, and this stochasticity gives rise to different exploratory behaviours. We note that if the agent adopts a fixed policy π, the MDP reduces to a Markov reward procss with induced state transition probability:

$$\mathcal{P}^{\pi}(s_{t+1}|s_t) = \sum_{a_t} \mathcal{P}(s_{t+1}|a_t, s_t)\pi(a_t|s_t). \tag{2.3}$$

The goal of reinforcement learning agent is to learn a policy that maximizes the expected cumulative return, possibly discounted as described above. A well-established approach for finding optimal policies is the *policy iteration* method (see Figure 2.1), which consists of two main steps: i) *policy evaluation* and ii) *policy improvement*. In the policy evaluation step, the RL agent computes the expected cumulative return for its current policy π, as a function of the state. This is called the *value function* V^{π} (Sutton and Barto, 1998; Szepesvári, 2010). In many problems of interest, the MDP M is unknown, but the agent has access to trajectories produced by interacting with M, which allow it to estimate V^{π} approximately. Once the value function V^{π} is estimated, the RL agent improves the policy π, modifying it into a new policy π', such that the expected return $V^{\pi'}(s) \geq V^{\pi}(s)$ for all $s \in \mathcal{S}$. The policy iteration cycle (Figure 2.1) continues until the optimal policy π^* is obtained, corresponding to optimal value function $V^{\star}(s)$ for all $s \in \mathcal{S}$. The existence of a unique $V^{\star}$ has been proven for MDPs with finite state and action spaces by Bellman (Bellman, 1957).

2.1.1 Policy Evaluation

In the policy evaluation step, the value function for the current policy is computed. The expected return when starting from state s and following policy π thereafter is the *state-value function*, $V^\pi : \mathcal{S} \to \mathbb{R}$, defined as,

$$\forall s \in \mathcal{S}, \;\; V^\pi(s) := \mathbb{E}_{\mathcal{P},\pi}\left[G_0 | s_0 = s\right]. \tag{2.4}$$

In other words, the value function evaluates the expected return of the RL agent's policy for each possible state. Additionally, one can compute the expectation of V^π over the initial state distribution p_0. This allows the comparison of policies when the state space is infinite as well.

Similarly, the *action-value function*, $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, estimates the expected value of the return when the agent starts in state s, takes action a, and follows policy π afterwards:

$$Q^\pi(s,a) = \mathbb{E}_{\mathcal{P},\pi}\left[G_0 | s_0 = s, a_0 = a\right]. \tag{2.5}$$

The ultimate goal of reinforcement learning agents is to learn a policy that maximizes value either at all states (in a finite MDP) or for a given start state distribution p_0.

Policy evaluation is the problem of obtaining (an approximation to) the value function of a Markov reward process defined by MDP M and a given policy π (Sutton and Barto, 1998; Szepesvári, 2010). Policy evaluation algorithms usually start with the following re-framing of the value function of a state in terms of the values of its successors:

$$\forall s \in \mathcal{S}, \;\; V^\pi(s) := \mathbb{E}_{\mathcal{P},\pi}\left[r(s_t, a_t) + \gamma V^\pi(s_{t+1}) | s_t = s\right], \tag{2.6}$$

which is also called Bellman equation for policy evaluation. If the state space is finite and the reward function and transition model of the MDP M are known, this can be re-written as a linear system of equations, and solved exactly or iteratively, as detailed below.

Let $N = |\mathcal{S}|$. Let $\boldsymbol{V}^\pi$ denote the N dimensional vector such that its coordinate i contains the state-value estimate of $V^\pi(i), i \in \mathcal{S}$ for a fixed order of the states. The Bellman equation (2.6) can be seen as the fixed point of the affine transformation $T^\pi : \mathbb{R}^N \to \mathbb{R}^N$ known as the Bellman operator (Szepesvári, 2010; Puterman, 2014) and define as:

$$T^\pi \boldsymbol{V}^\pi := \boldsymbol{R}^\pi + \gamma \boldsymbol{P}^\pi \boldsymbol{V}^\pi. \tag{2.7}$$

Here, $\boldsymbol{R}^{\pi} \in \mathbb{R}^{N}$ denotes the reward vector for policy π, whose ith coordinate is $\boldsymbol{R}^{\pi}_{i} := \mathbb{E}_{\pi}\left[r(s^{i}, .)\right]$ and the elements of the transition matrix $\boldsymbol{P}^{\pi}$ are as defined in 2.3. The value function can then be computed as the fixed point of T^{π}:

$$T^{\pi}\boldsymbol{V}^{\pi} = \boldsymbol{V}^{\pi}. \tag{2.8}$$

Note that T^{π} is a contraction mapping due to the fact that $\gamma < 1$, which together with some mild assumptions on the state visitation Markov chain guarantees that the fixed point exists and is unique (Szepesvári, 2010).

However, in many cases of interest, M is unknown, but we have access to trajectories containing state transitions and immediate rewards sampled from π, a case called *on-policy policy evaluation.* In this case, the right-hand-side of 2.6 can be approximated from samples, leading to Monte Carlo or temporal-difference (TD) learning methods.

When the state space of M is relatively small, tabular methods that represent the value of each state can be used individually. However, in problems with large (or even continuous) state spaces, parametric representations for the value function are typically needed in order to defeat the curse of dimensionality and exploit the fact that similar states will have similar values. In this thesis, we focus on policy evaluation with linear function approximation in the batch setting, where we have access to a set of trajectories sampled from the target policy or the behaviour policy. The problem of policy evaluation splits into two categories of *on-policy estimation*, when the target policy and behaviour policies are the same and *off-policy estimation* when the target policy and behaviour policy are not the same. In this thesis in Chapters 3 and 4 we choose to adopt the on-policy setting and in Chapter 5 we adopt the off-policy setting. Future rewards play fundamental role on the state value and this dependency diminishes as the value of discount factor γ approaches to zero, in which the problem of state value estimation reduces to standard supervised learning setting. One standard way of computing state-value function (2.4) is the recursive representation of Equation (2.2) for two successive time-steps.

In real-world applications the state-space is either large or infinite. This gives rise to an inherent problem in stat-value estimation, in which the size of state space increase exponentially as the number of state variables increases. This problem is known as the curse of dimensionality. This pushes us towards the approximation of state-value function. In this thesis, we use the most common approximation approach, known as a linear function approximation with the parameter vector $\theta \in \mathbb{R}^{d}$.

Let $\Phi \in \mathbb{R}^{\mathcal{S} \times d}$ be a feature representation that associates each state $s \in \mathcal{S}$ to a d-dimensional feature vector $\phi_s^\top = \Phi(s,:) \in \mathbb{R}^d$ with $d \ll |\mathcal{S}|$. The goal in the linear function approximation is to find a parameter vector $\theta \in \mathbb{R}^d$ such that $\hat{V}^\pi = \Phi\theta$ is a good approximation to V^π. To do so, we assume that we have access to a collection $X = (x_1, \ldots, x_m)$ of finite trajectories sampled from M by π, where each x_i is a sequence of states, actions and rewards. A dataset $X \in \mathbb{X}^m$ is considered as a multiset of m i.i.d. trajectories and $\tau \in \mathbb{X}$ be a trajectory chosen from $\mathbb{X}$. Each trajectory $\tau = \{(S_i^\tau, A_i^\tau, R_i^\tau)\}_{t=0}^{l_\tau}$ starts with a state drawn w.r.t. the initial state distribution ρ, collects transition triples based on the fixed policy π and terminates with an absorbing state from the set of absorbing states $\mathcal{B}$.

2.2 Batch Off-policy Deep Reinforcement Learning

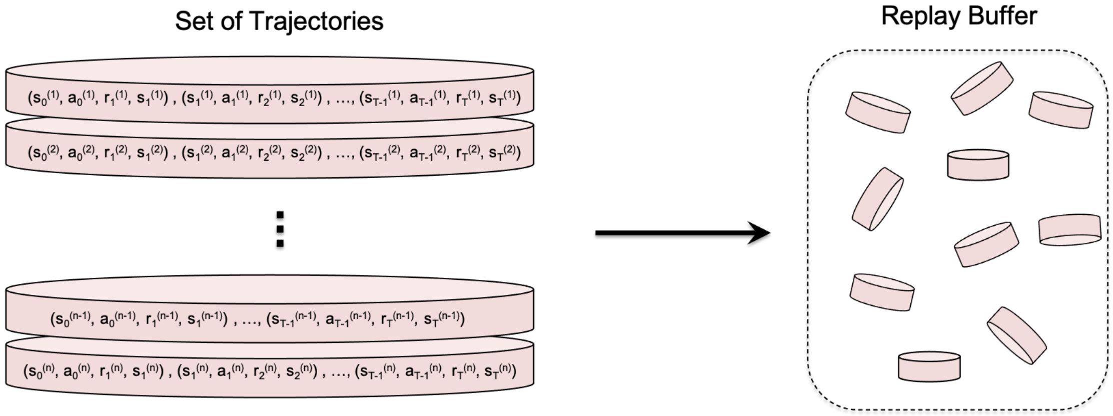

Set of Trajectories
Replay Buffer
(s0(1), a0(1), r1(1), s1(1)) , (s1(1), a1(1), r2(1), s2(1)) , ..., (sT-1(1), aT-1(1), rT(1), sT(1))
(s0(2), a0(2), r1(2), s1(2)) , (s1(2), a1(2), r2(2), s2(2)) , ..., (sT-1(2), aT-1(2), rT(2), sT(2))
(s0(n-1), a0(n-1), r1(n-1), s1(n-1)) , ..., (sT-1(n-1), aT-1(n-1), rT(n-1), sT(n-1))
(s0(n), a0(n), r1(n), s1(n)) , (s1(n), a1(n), r2(n), s2(n)) , ..., (sT-1(n), aT-1(n), rT(n), sT(n))

FIGURE 2.2: A schematic of replay buffer mechanism. Replay buffer (*on the right*) receives a set of n trajectories (*on the left*), each formed by the concatenation of correlated tuples $(s_t^j, a_t^j, r_{t+1}^j, s_{t+1}^j)$, where $j = 1, 2, \ldots, n$ denotes the trajectory number and $t = 0, 1, \ldots, T-1$ is the tuple index in each trajectory. The replay buffer subsequently breaks each trajectory into its constituent tuples and stores the resulting tuples from the decorrelated trajectories. Note that the small disks in the replay buffer on the right represent tuples.

In batch off-policy setting, the reinforcement learning agent decouples the data collection phase from the policy training phase (*i.e.* off-policy), which ensures that the learning system is not tied to a particular exploration algorithm and it also ensures disjointedness between the training datasets provided for the RL algorithm in different settings. This feature is essential in real-world applications, where the training data is provided privately for the reinforcement learning model by a certain industry, which is subsequently used to train the private model.

For more clarification, let us walk through an illustrative example. In the healthcare industry, the system's goal is to design medical treatment policies, through which at each stage patients receive different treatment recommendations based on the information obtained from the patients' history of interaction with the system. In such instances, it is often hazardous to train and employ the reinforcement learning models simultaneously. Thus, the reinforcement learning agent is first trained with a batch of patients' treatment records (trajectories) in off-policy mode, and the trained policy is subsequently released for decision making.

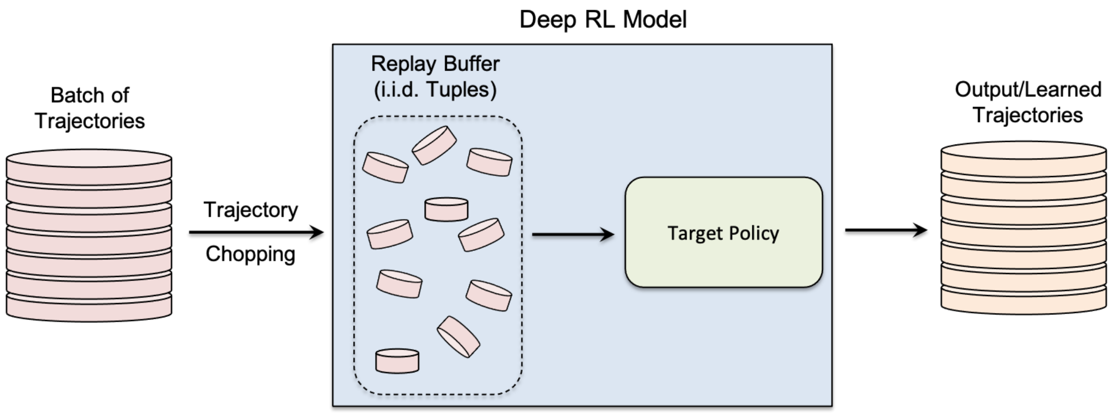


FIGURE 2.3: Batch off-policy deep reinforcement learning architecture. An external behaviour policy generates a batch of i.i.d. trajectories composed of transition tuples (state, action, reward, new state), which are delivered to the deep RL model. The replay buffer mechanism as an internal part of the model decorrelates each trajectory into a collection of i.i.d. transition tuples and then uses them in the form of mini-batches to train the target policy.

Reinforcement learning agents do not have knowledge of the environment at the very initial stage of learning, and acquire the necessary experience through continued interactions with the environment. A reinforcement learning agent can acquire the necessary information in two ways: *on-policy* and *off-policy*. In the off-policy setting, the agent collects the necessary information via the *behaviour policy* π_b (exploration), and subsequently uses the acquired data to train the *target policy* π_f (exploitation). Figure 2.3 presents a schematic of off-policy deep RL architecture. In the on-policy setting, on the other hand, the agent uses the target policy that is trained so far to obtain data by interacting with the environment. From the privacy point of view, since the private data is assumed to exist a priori, off-policy methods are natural choices to analyze in this regard.

The reinforcement learning agent requires deep neural networks as non-linear function approximators to train the target policy in complex environments. Most of the existing off-policy deep reinforcement learning methods such as *Deep Deterministic Policy Gradients* (DDPG) (Lillicrap et al., 2015), *Soft Actor Critic* (SAC) (Haarnoja et al., 2018), and *Deep Q-learning algorithm* (DQN) (Mnih et al., 2015) are indeed considered as near-on-policy algorithms since their exploration policy is greatly correlated with the learning policy, yet they are labelled as an off-policy algorithm due to their use of classic off-policy Q-learning (Watkins and Dayan, 1992). Thus, decoupling exploration from the learning phase becomes challenging in the methods mentioned above. To properly address the vulnerability of deep reinforcement learning algorithms to membership inference attacks (MIAs), we need to adopt a truly off-policy deep reinforcement learning method that explicitly decouples exploration and learning steps. The state-of-the-art off-policy model that is widely used as the basis of other deep RL algorithms is the Batch-Constrained deep Q-learning (BCQ) (Fujimoto et al., 2019) method. Structurally, BCQ trains a generative model on the input trajectories such that the model learns the relationship between the visited states in the input trajectories and the corresponding taken actions. The BCQ algorithm subsequently uses the developed generative model to train a deep Q-network, which ultimately learns to sample the highest valued actions similar to the ones in the input trajectories.

The fact that the input trajectories in off-policy deep RL models are temporally correlated necessitates the use of a mechanism that converts the input data to *i.i.d.* samples before passing it to the deep network. A widespread and fundamental data management mechanism that has become an inevitable part of the existing off-policy deep reinforcement learning models is *experience replay buffer* or *replay buffer*. Application of experience replay buffer significantly improves the sample efficiency and stability of off-policy deep reinforcement learning algorithms (Mnih et al., 2015). The concept of replay buffer was first introduced in Lin (1992). Several years later, authors in Mnih et al. (2015) for the first time designed an off-policy deep reinforcement learning algorithm that incorporates replay buffer in its architecture. The main intuition behind the application of replay buffer in deep reinforcement learning lies at the heart of reinforcement learning theory. It is well-studied that the Q-learning algorithm easily diverges in the case of linear function approximation (Sutton and Barto, 1998). The solution that replay buffer offers to the problem of the divergence of Q-learning algorithm is to decorrelate the input trajectories and subsequently treat each transition tuple as an i.i.d. sample point (Figure 2.2). This intermediate decorrelation step significantly impacts data efficiency and helps the deep reinforcement learning algorithm converge to the optimal policy according to the law of large numbers. Moreover, it allows the deep RL algorithm to benefit from mini-batch training and shuffling techniques, which are proven to improve the performance of deep reinforcement learning algorithms significantly (Mnih et al., 2015; Zhang and Sutton, 2017; Silver et al., 2017; Liu and Zou, 2018; Fedus et al., 2020).

In the context of RL, a data point in a batch of data is a sequence (a trajectory) of observations, actions and any kind of information that the RL agent exchange with the environment, which is denoted as the following,

$$\tau_T = (s_0, a_0, r_1, s_1), (s_1, a_1, r_2, s_2), \ldots, (s_{T-1}, a_{T-1}, r_T, s_T). \tag{2.9}$$

The RL agent receives an input batch of data in the form of trajectories provided by an exploratory agent, and subsequently uses this data to train the target policy. The output of the trained target policy consists of data points (trajectories) produced via interaction between the target policy and the environment (Figure 2.3).

2.3 Differential privacy

Differential Privacy (DP) has become a de-facto standard for designing privacy-preserving machine learning algorithms with strong privacy guarantees Dwork et al. (2014); Xiao et al. (2010); Mohammed et al. (2011); Chen et al. (2011, 2014). DP takes a user-centric approach, by providing privacy guarantees based on the difference of the outputs of a learning algorithm trained on two databases differing in a single user. The central goal is to bound the loss in privacy that a user can suffer when the result of an analysis on a database with her data is made public. This can incentivize users to participate in studies using sensitive data, e.g. mining of medical records. In the context of machine learning, differentially private algorithms are useful because they allow learning models in such a way that their parameters do not reveal information about the training data (McSherry and Talwar, 2007). For example, one can think of using historical medical records to learn prognostic and diagnostic models which can then be shared between multiple health service providers without compromising the privacy of the patients whose data was used to train the model.

To formalize the above discussion, let $\mathcal{X}$ be an *input space* and $\mathcal{Y}$ an *output space*. Suppose A is a randomized algorithm that takes as input a tuple $X = (x_1, \ldots, x_m)$ of elements from $\mathcal{X}$ for some $m \geq 1$ and outputs a (random) element $A(X)$ of $\mathcal{Y}$. We interpret $X \in \mathcal{X}^m$ as a dataset containing data from m individuals and define its *neighbouring* datasets as those that differ from X in their last element: $X' = (x_1, \dots, x_{m-1}, x'_m)$ with $x_m \neq x'_m$. Formally, we should define neighbouring datasets as those which differ in one element, not necessarily the last. But we are implicitly assuming here that the order of the elements in X does not affect the distribution of $A(X)$, so we can assume without loss of generality that the difference between neighbouring datasets is always in the last element. We denote this (symmetric) relation by $X \simeq X'$. A is (ε, δ)*-differentially private* for some $\varepsilon, \delta > 0$ if for every $m \geq 1$, every pair of datasets $X, X' \in \mathcal{X}^m$, $X \simeq X'$, and every measurable set $\Omega \subseteq \mathcal{Y}$ we have

$$\mathbb{P}[A(X) \in \Omega] \leq e^{\varepsilon}\mathbb{P}[A(X') \in \Omega] + \delta \ . \tag{2.10}$$

This definition means that the distribution over possible outputs of A on inputs X and X' is very similar, so revealing this output leaks almost no information on whether x_m or x'_m was in the dataset.

2.3.1 Output perturbation mechanism

A simple yet popular way to design a DP algorithm for a given function $f : \mathcal{X}^m \to \mathcal{Y}$ is the *output perturbation* mechanism, which releases $A(X) = f(X) + \eta$, where η is noise sampled from a properly calibrated distribution. For real outputs $\mathcal{Y} = \mathbb{R}^d$, the Laplace (resp. Gaussian) mechanism (see e.g. Dwork et al. (2014)) samples each component of the noise $\eta = (\eta_1, \dots, \eta_d)$ i.i.d. from a Laplace (resp. Gaussian) distribution with standard deviation $O(\mathrm{GS}_1^{(m)}(f)/\varepsilon)$ (resp. $O(\mathrm{GS}_2^{(m)}(f)\ln(1/\delta)/\varepsilon)$), where $\mathrm{GS}_p^{(m)}(f)$ is the *global sensitivity* of f defined in the following definition.

Definition 2.1 (Global Sensitivity (Dwork et al., 2006))**.**

$$\mathrm{GS}_p^{(m)}(f) = \sup_{X,X' \in \mathcal{X}^m, X \simeq X'} \|f(X) - f(X')\|_p \ ,$$

where $x \simeq x'$ means that x' can be obtained by replacing a single entry in x.

The notion of *global sensitivity* captures the maximum impact on the output of a deterministic algorithm induced by changing a single input entry, we generalize this notion in the following definition.

Definition 2.2 (Global Sensitivity Generalized)**.** Let $A = \{A_n\}_{n\in\mathbb{N}}$ be a family of deterministic algorithms of the form $A_n : \mathbb{X}^n \to \mathbb{R}^d$. The *global sensitivity* of A is defined as,

$$\mathrm{GS}_p^{(n)}(A) := \sup_{x,x'\in\mathbb{X}^n : x\sim x'} \|A(x) - A(x')\|_p = \left(\sup_{x,x'\in\mathbb{X}^n : x\sim x'} \|A(x) - A(x')\|_p^p \right)^{1/p}, \tag{2.11}$$

where $x \simeq x'$ means that x' can be obtained by replacing a single entry in x.

Calibrating noise to the global sensitivity is a worst-case approach that requires taking the supremum over all possible pairs of neighbouring datasets, and in general, does not account for the fact that in some datasets, privacy can be achieved with substantially smaller perturbations. Notably, in reinforcement learning applications where data efficiency is of great importance (e.g. healthcare applications), one needs to adopt data-dependent notions of sensitivity for better utility-privacy tradeoff management. In Chapter 3 we adopt a notion of sensitivity that is suitable for policy evaluation algorithms.

2.3.2 Private Reinforcement Learning

Contrary to supervised learning, literature on privacy-preserving RL is quite scarce. Balle et al. (2016) is the first study that addressed privacy in general RL. Prior to this work the focus was on privacy-preserving algorithms with partial feedback, e.g. bandit-type problems (Mishra and Thakurta, 2015; Tossou and Dimitrakakis, 2016). The study on privacy-preserving bandit algorithms is still an active research direction (Shariff and Sheffet, 2018; Basu et al., 2019; Dubey, 2021).

In the control setting, Wang and Hegde (2019) propose a privacy-preserving Q-learning algorithm in the continuous space RL setting using functional noise (Hall et al., 2013). Their theoretical analysis proposed in Wang and Hegde (2019) provides useful insight on the interplay between the utility and privacy of the proposed algorithm. In a similar line of research, Vietri et al. (2020) employed the relaxed notion of joint differential privacy (JDP) in the tabular setting and developed a private optimism-based learning algorithm with PAC and regret bounds. Vietri et al. (2020) provide the first formal utility guarantee of the privacy-preserving Upper Confidence Bound (PUCB) algorithm in the tabular setting. Using the private UCB algorithm proposed in Vietri et al. (2020), authors propose an algorithm that computes the optimistic private Q-function. The paper leaves the extension of proposed algorithms to non-tabular setting to the future work.

Our principal motivation for studying differential privacy (DP) in reinforcement learning (RL) comes from medical applications. Learning and evaluation of dynamic treatments is a crucial part of personalised medicine and adaptive clinical trials (Kulynych and Greely, 2017; Stiles and Appelbaum, 2019). Privacy at the level of trajectories is a natural requirement in these applications because each trajectory represents the evolution of a single patient during their treatment. We will include this example in the introduction to clarify the motivation of our work.

Chapter 3 provides the first study of privacy-preserving policy evaluation along with the comprehensive study of privacy-utility trade-off in the linear function approximation setting. Research on privacy-preserving policy evaluation algorithms is still at its infant stages and to the best of our knowledge algorithms proposed in this thesis are the only privacy-preserving algorithms that are particularity tailored to the policy evaluation setting. Our rationale for going in this direction was that policy evaluation is a building block of many RL algorithms, and at the same time provides an easier scenario than the control case. Hence, theoretical guarantees can be easier to obtain and the privacy-utility trade-off is crisper. We also wanted to work in a setting that is independent of the exploration algorithm, as the latter is still a very active area of research in RL even outside of any privacy considerations.

2.4 Membership Inference Attacks

Membership Inference Attacks (MIAs) are a form of adversarial attacks that use a publicly available trained model, and some other auxiliary information, in order to identify whether a particular sample point was used in the training of the model or not. This is relevant, for example, if we consider the healthcare setting, because knowing if a person was, for example, included in training a predictor for disease progression conditioned on treatment can indicate their condition, which is sensitive private information.

In the research on MIAs, one takes the perspective of an attacker and tries to design some approach by which membership of a given data point into the training data of a model from a particular class can be established. Often, classifiers are used as a tool to learn how to perform this identification. MIAs can be either black-box or white-box. In black-box MIAs, the attacker can query the model by using its own data but does not have any knowledge of how the model was trained. In the white-box setting, further information about the model, for example the training algorithm, may be available. While white-box

MIAs can be a lot more effective, they are also more difficult to implement in practice, as they require particular kinds of extra information, and therefore they are less relevant. Hence, we will restrict our attention to black-box MIAs. In this section, we provide a brief overview of relevant work in MIA design in both supervised learning and RL settings.

2.4.1 Membership inference in supervised machine learning.

There exists an extensive body of literature on membership inference attacks on supervised machine learning models, where they provide an interesting counterpart to privacy analyses. In the context of genomics data, authors in Dwork et al. (2015) propose membership inference attacks (tracing attacks) against supervised classifiers. The main intuition in the design of membership inference attacks is that a publicly available trained model h will exhibit higher confidence in the individuals who participated in the training data, and thus the instances in the training are more vulnerable to privacy threats.

The *shadow model* technique (Shokri et al., 2017) is known as an effective and practical approach for designing membership inference attack models. Shadow models are parallel models that can be trained by an attacker on data sets often sampled from the same distribution as the underlying distribution of the private data. In this method, the adversary trains the models with complete knowledge of the training set. Thus, using auxiliary membership information and the trained shadow models, the adversary can build a membership classifier that identifies whether an individual has participated in the training of similarly trained models.

In the rest of this section, we provide a brief overview of the existing studies that adopt the shadow model technique as their principle attack training strategy in the supervised setting, and we refer the interested reader to a comprehensive survey on membership attacks in machine learning by Hu et al. (2021) for further information.

Authors in Shokri et al. (2017) for the first time adopted shadow model training technique in the supervised setting and proposed a membership attack against a deep classifier in the black-box setting. (Shokri et al., 2017) used shadow model training to design membership inference attacks in black-box settings by replicating the behaviour of the target model through training shadow models on some public datasets drawn from the same distribution as the private dataset used to train the target model. Salem et al. (2019) discuss two main assumptions adopted by Shokri et al. (2017) and propose attack strategies with milder assumptions. They test the performance of the proposed attacks on some datasets such as MNITS, CIFAR100, and Purchase-10. The authors of Yeom et al. (2018) show that overfitting is sufficient for the adversary to perform membership inference attacks against several machine learning models, such as regression and deep convolutional neural networks (CNNs). Many other works study membership attacks in machine learning in different settings and under different sets of assumptions (Long et al., 2017; Hayes et al., 2019a; Sablayrolles et al., 2019; Long et al., 2018).

2.4.2 Membership inference attack in reinforcement learning

The most suitable setting to test the vulnerability of deep RL algorithms against MIAs is the batch off-policy setting. In the off-policy batch reinforcement learning, the training set x for a Deep RL agent is composed of m trajectories $\{\tau_1, ..., \tau_m\}$, each represents a data-point and is composed of a finite number of tuples in the form of $\langle s_t, a_t, r_t, s_{t+1}\rangle$. A trajectory as a single data point in a dataset can be interpreted differently. For example, a trajectory can capture a patient's interactions with different parts of a hospital, from the reception office to the patient release office. Trajectories can have various finite sizes.

In deep RL, adopting a similar MIA design principle as in a supervised setting arises some non-trivial and fundamental challenges. The first challenge is that a data point in a deep RL setting is a complete trajectory composed of temporally correlated entry points, and this gives rise to a much richer input representation compare to the classic supervised learning to set. Exploring the inherent correlation within the input data in a deep RL model is considered as one of the main sources of high variance in model outputs Sutton and Barto (1998). The second challenge arises from the inevitable use of the data transformation mechanism termed *replay buffer* Mnih et al. (2015). Replay buffer is mainly used to remove temporal correlation within a trajectory and smooth out the distributional discrepancy at the trajectory. This intermediate transformation phase adds a new source of noise to the input data from the attacker perspective, which is while improving on the stability of the target model predictions, it also makes the membership inference a more challenging task against deep RL models. Investigating the trade-off between the initial correlation among transition tuples that potentially helps the attacker to infer the membership of the target trajectory in the training set and the existence of replay buffer is one of the main contributions of this paper. This trade-off, of course, is a subtle and non-trivial trade-off for a non-RL audience. The third challenge is due to the temporal nature of model training in deep RL, which gives rise to a fundamentally different design of shadow training technique that is discussed in more detail in Section 5.5.

Pan et al. (2019) propose a shadow training model to infer the transition model used to train the target policy from the set of candidate transition dynamics. In such a setting, the set of transition models act as the set of classes in standard classification tasks. The assumption of having access to a collection of transition dynamics is infeasible to many real-wold RL settings and less appealing to the industrial audience, where the concern is the privacy of individuals who participated in a particular study or training. To the best of our knowledge, our study (Chapter 5) is the first work that addresses the problem of membership inference attack in reinforcement learning where the target model is trained on the environment accessible to the adversary with the same query access level as the target model.

2.4.3 Performance Metrics

We adopt the standard performance metrics used in the classification literature (Sokolova and Lapalme, 2009) to evaluate the performance of our proposed attack models against the deep reinforcement learning model. We measure the performance of the attack classifier as a function of the following quantities:

1. **True Positives** (TP): Number of correctly recognized positives,
2. **True Negatives** (TN): Number of correctly recognized negatives,
3. **False Positives** (FP): Number of incorrectly recognized positives,
4. **False Negatives** (FN): Number of incorrectly recognized negatives.

We use the following metrics to analyze our data:

Overall accuracy **(ACC)** captures the overall performance of attack classifier and is calculated as follows,

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{2.12}$$

Precision **(PR)** shows the fraction of pairs classified as matching pairs that are indeed coming from the same model, and is written as,

$$\mathrm{PR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{2.13}$$

Recall **(RE)** measures the fraction of matching pairs that the attack classifier can infer correctly, and is computed as follows

$$\mathrm{RE} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{2.14}$$

The evaluation metrics mentioned above are sometimes misleading. For instance, accuracy is a metric that heavily depends on the distribution of the classifier input pairs. For instance, where 90% of input pairs are negative and 10% are positive, a simple attack model that without any learning outputs ’negative’ for any test pairs, exhibits 90% accuracy. Thus, to evaluate the performance of our proposed attack models and improve the robustness of our findings, we further employ two other evaluation metrics:

F1 score **(F1)** is the harmonic mean of the precision (PR) and recall (RE), found as

$$\mathrm{F1} = \left(\frac{1}{2}\left(\frac{1}{\mathrm{PR}} + \frac{1}{\mathrm{RE}}\right)\right) = \frac{2.\mathrm{PR}.\mathrm{RE}}{\mathrm{PR} + \mathrm{RE}} \tag{2.15}$$

Matthews Correlation Coefficient (MCC) Matthews (1975) calculates the correlation between the predicted and the true classification labels, and is defined as

$$\mathrm{MCC} = \frac{\mathrm{TP}.\mathrm{TN} - \mathrm{FP}.\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}. \tag{2.16}$$

MCC is an effective and meaningful combination of all four quantities TP, TN, FP, and FN, which ranges from -1 to 1. The closer MCC is to 1, the better the model performs (Chicco et al., 2021). MCC$=0$ shows that the model is a random guesser. The other evaluation metrics ACC, PR, RE, and F1 vary in the range $[0,1]$. In a well-performing model, all of these evaluation metrics have values close to 1.

Finally, to show the performance of our proposed MIA classifiers in individual and collective modes at different classification thresholds θ, we plot receiver operating characteristic (ROC) curve, which shows the changes of RE as a function of *False Positive Rate* (FPR),

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{2.17}$$

for different values of θ.

Chapter 3

Private Monte Carlo Policy Evaluation

In this chapter, we present the first differentially private algorithms for reinforcement learning, which apply to the task of evaluating a fixed policy. We establish two approaches for achieving differential privacy, provide a theoretical analysis of the privacy and utility of the two algorithms, and show promising results on simple empirical examples.

3.1 Introduction and motivation

Learning how to make decisions under uncertainty is becoming paramount in many practical applications, such as medical treatment design, energy management, adaptive user interfaces, recommender systems etc. Reinforcement learning (Sutton and Barto, 1998) provides a variety of algorithms capable of handling such tasks. However, in many practical applications, aside from obtaining good predictive performance, one might also require that the data used to learn the predictor be kept confidential. This is especially true in medical applications, where patient confidentiality is very important, and in other applications which are user-centric (such as recommender systems). *Differential privacy* (DP) (Dwork, 2006) is a very active research area, originating from cryptography, but which has now been embraced by the machine learning community. DP is a formal model of privacy used to design mechanisms that reduce the amount of information leaked by the result of queries to a database containing sensitive information about multiple users (Dwork, 2006). Many supervised learning algorithms have differentially private versions, including logistic regression (Chaudhuri and Monteleoni, 2009; Chaudhuri et al., 2011), support vector machines (Chaudhuri et al., 2011; Rubinstein et al., 2012; Jain and Thakurta, 2013), and the lasso (Thakurta and Smith, 2013). However, differential privacy for reinforcement learning tasks has not been tackled yet, except for the simpler case of bandit problems (Shariff and Sheffet, 2018; Mishra and Thakurta, 2015; Tossou and Dimitrakakis, 2016).

In this chapter, we tackle differential privacy for reinforcement learning algorithms for the full Markov Decision Process (MDP) setting. We develop differentially private algorithms for the problem of policy evaluation, in which a given way of behaving has to be evaluated quantitatively. We start with the batch, first-visit Monte Carlo approach to policy evaluation, which is well understood and closest to regression algorithms, and provide two differentially private versions, which come with formal privacy proofs as well as guarantees on the quality of the solution obtained. Both algorithms work by injecting Gaussian noise into the parameters vector for the value functions, but they differ in the definition of the noise amount. Our privacy analysis techniques are related to previous output perturbation for empirical risk minimization (ERM), but there are some domain specific challenges that need to be addressed. Our utility analysis identifies parameters of the MDP that control how easy it is to maintain privacy in each case. The theoretical utility analysis, as well as some illustrative experiments, show that the accuracy of the private algorithms does not suffer (compared to usual Monte Carlo) when the data set is large.

The rest of this chapter is organized as follows. In Section 3.2 we provide background notation and results on differential privacy and Monte Carlo methods for policy evaluation. Section 3.3 presents our proposed algorithms. The privacy analysis and the utility analysis are outlined in Section 3.4 and Section 3.5 respectively. Detailed proofs for both of these sections are given in the Supplementary Material. In Section 3.8 we provide empirical illustrations of the scaling behaviour of the proposal algorithms, using synthetic MDPs, which try to mimic characteristics of real applications. Finally, we conclude in Section 3.9 with a discussion of related work and avenues for future work.

3.2 Background and notation

In this section we provide background on differential privacy and policy evaluation from Monte Carlo estimates.

3.2.1 Monte Carlo Policy Evaluation

We will use a Monte Carlo approach, in which the returns of the trajectories in X are used as regression targets to fit the parameters in $\hat{V}^\pi$ via a least squares approach (Sutton and Barto, 1998). In particular, we consider first-visit Monte Carlo estimates obtained as follows. Suppose $x = ((s_1, a_1, r_1), \ldots, (s_T, a_T, r_T))$ is a trajectory that visits s and $i_{x,s}$ is the time of the first visit to s; that is, $s_{i_{x,s}} = s$, and $s_t \neq s$ for all $t < i_{x,s}$. The return collected from this first visit is given by

$$F_{x,s} = \sum_{t=i_{x,s}}^{T} r_t \gamma^{t-i_{x,s}} = \sum_{t=0}^{T-i_{x,s}} r_{t+i_{x,s}} \gamma^t \ ,$$

and provides an unbiased estimate of $V^\pi(s)$. For convenience, when state s is not visited by trajectory x we assume $F_{x,s} = 0$.

Given the returns from all first visits corresponding to a dataset X with m trajectories, we can find a parameter vector for the estimator $\hat{V}^\pi$ by solving the optimization problem $\arg\min_\theta J_X(\theta)$, where

$$J_X(\theta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{s \in \mathcal{S}_{x_i}} \rho_s (F_{x_i,s} - \phi_s^\top \theta)^2 \ , \tag{3.1}$$

and $\mathcal{S}_x$ is the set of states visited by trajectory x. The regression weights $0 \leq \rho_s \leq 1$ are given as an input to the problem and capture the user's believe that some states are more relevant than others. It is obvious that $J_X(\theta)$ is a convex function of θ. However, in general it is not strongly convex and therefore the optimum of $\arg\min_\theta J_X(\theta)$ is not necessarily unique. On the other hand, it is known that differential privacy is tightly related to certain notions of stability (Thakurta and Smith, 2013), and optimization problems with non-unique solutions generally pose a problem to stability. In order to avoid this problem, the private policy evaluation algorithms that we propose in Section 3.3 are based on optimizing slightly modified versions of $J_X(\theta)$ which promote stability in their solutions. Note that the notions of stability related to DP are for worst-case situations: that is, they need to hold for every possible pair of neighbouring input dataset $X \simeq X'$, regardless of any generative model assumed for the trajectories in those datasets. In particular, these stability considerations are not directly related to the variance of the estimates in $\hat{V}^\pi$.

We end this section with a discussion of the main obstruction to stability, i.e. the cases where $\arg\min_\theta J_X(\theta)$ fails to have a unique solution. Given a dataset X with m trajectories we define a vector $F_X \in \mathbb{R}^{\mathcal{S}}$ containing the average first visit returns from all trajectories in X that visit a particular state. In particular, if X_s represents the multiset of trajectories from X that visit state s at some point, then we have

$$F_X(s) = F_{X,s} = \frac{1}{|X_s|} \sum_{x \in X_s} F_{x,s} \ . \tag{3.2}$$

If s is not visited by any trajectory in X we set $F_{X,s} = 0$. To simplify notation, let $F_X \in \mathbb{R}^{\mathcal{S}}$ be the vector collecting all these estimates. We also define a diagonal matrix $\Gamma_X \in \mathbb{R}^{\mathcal{S} \times \mathcal{S}}$ with entries given by the product of the regression weight on each state and the fraction of trajectories in X visiting that state: $\Gamma_X(s,s) = \rho_s |X_s|/m$. Solving for θ in $\nabla_\theta J_X(\theta) = 0$, it is easy to see that any optimal $\theta_X \in \arg\min_\theta J_X(\theta)$ must satisfy

$$\Phi^\top \Gamma_X \Phi \theta_X = \Phi^\top \Gamma_X F_X \ . \tag{3.3}$$

Thus, this optimization has a unique solution if and only if the matrix $\Phi^\top \Gamma_X \Phi$ is invertible. Since it is easy to find neighbouring datasets $X \simeq X'$ where at most one of $\Phi^\top \Gamma_X \Phi$ and $\Phi^\top \Gamma_{X'} \Phi$ is invertible, optimizing $J_X(\theta)$ directly poses a problem to the design differentially private policy evaluation algorithms with small perturbations. Next we present two DP algorithm based on stable policy evaluation algorithms.

3.3 Private First-Visit Monte Carlo Algorithms

In this section we give the details of two differentially private policy evaluation algorithms based on first-visit Monte Carlo estimates. Each of these algorithms corresponds to a different stable version of the minimization $\arg\min_\theta J_X(\theta)$ described in previous section. A formal privacy analysis of these algorithms is given in Section 3.4. Bounds showing how the privacy requirement affects the utility of the value estimates are presented in Section 3.5.

3.3.1 Algorithm DP-LSW

One way to make the optimization $\arg\min_\theta J_X(\theta)$ more stable to changes in the dataset X is to consider a similar least-squares optimization where the optimization weights do not change with X, and guarantee that the optimization problem is always strongly convex. Thus, we consider a new objective function given in terms of a new set of positive regression weights $w_s > 0$. Let $\Gamma \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}$ be a diagonal matrix with $\Gamma(s,s) = w_s$. We define the objective function as:

$$J_X^w(\theta) = \sum_{s\in\mathcal{S}} w_s (F_{X,s} - \phi_s^\top \theta)^2 = \|F_X - \Phi\theta\|_{2,\Gamma}^2 \ , \tag{3.4}$$

where $\|v\|_{2,\Gamma}^2 = \|\Gamma^{1/2}v\|_2^2 = v^\top \Gamma v$ is the weighted L_2 norm. To see the relation between the optimizations over J_X and J_X^w, note that equating the gradient of $J_X^w(\theta)$ to 0 we see that a minimum $\theta_X^w \in \arg\min_\theta J_X^w(\theta)$ must satisfy

$$\Phi^\top \Gamma \Phi \theta_X^w = \Phi^\top \Gamma F_X \ . \tag{3.5}$$

Thus, the optimization problem is well-posed whenever $\Phi^\top\Gamma\Phi$ is invertible, which henceforth will be our working assumption. Note that this is a mild assumption, since it is satisfied by choosing a feature matrix Φ with full column rank. Under this assumption we have:

$$\theta_X^w = \left(\Phi^\top \Gamma \Phi\right)^{-1} \Phi^\top \Gamma F_X = \left(\Gamma^{1/2}\Phi\right)^{\dagger} \Gamma^{1/2} F_X \ , \tag{3.6}$$

where $M^\dagger$ denotes the Moore–Penrose pseudo-inverse. The difference between optimizing $J_X(\theta)$ or $J_X^w(\theta)$ is reflected in the differences between (3.3) and (3.5). In particular, if the trajectories in X are i.i.d. and p_s denotes the probability that state s is visited by a trajectory in X, then taking $w_s = \mathbb{E}_X[\rho_s |X_s|/m] = \rho_s p_s$ yields a loss function $J_X^w(\theta)$ that captures the effect of each state s in $J_X(\theta)$ in the asymptotic regime $m \to \infty$. However, we note that knowledge of these visit probabilities is not required for running our algorithm or for our analysis.

Our first DP algorithm for policy evaluation applies a carefully calibrated output perturbation mechanism to the solution θ_X^w of $\arg\min_\theta J_X^w(\theta)$. We call this algorithm DP-LSW, and its full pseudo-code is given in Algorithm 1. It receives as input the dataset X, the regression weights w, the feature representation Φ, and the MDP parameters $R_{\max}$ and γ. Additionally, the algorithm is parametrized by the privacy parameters ε and δ. Its output is the result of adding a random vector η drawn from a multivariate Gaussian distribution $\mathcal{N}(0, \sigma_X^2 I)$ to the parameter vector θ_X^w. In order to compute the variance of η the algorithm needs to solve the discrete optimization problem $\psi_X^w = \max_{0\le k\le K_X} e^{-k\beta}\varphi_X^w(k)$, where $K_X = \max_{s\in\mathcal{S}} |X_s|$, β is a parameter computed in the algorithm, and $\varphi_X^w(k)$ is given by the following expression:

$$\varphi_X^w(k) = \sum_{s\in\mathcal{S}} \frac{w_s}{\max\{|X_s| - k, 1\}^2} \ . \tag{3.7}$$

Note that ψ_X^w can be computed in time $O(K_X N)$.

Algorithm 1 DP-LSW

```
Require: $X$, $\Phi$, $\gamma$, $R_{\max}$, $w$, $\varepsilon$, $\delta$
    Compute $\theta_X^w$                                                    ▷ cf. (3.6)
    Let $\alpha \leftarrow \frac{5\sqrt{2\ln(2/\delta)}}{\varepsilon}$ and $\beta \leftarrow \frac{\varepsilon}{4(d+\ln(2/\delta))}$
    Let $\psi_X^w \leftarrow \max_{0\le k\le K_X} e^{-k\beta}\varphi_X^w(k)$                ▷ cf. (3.7)
    Let $\sigma_X \leftarrow \frac{\alpha R_{\max}\|(\Gamma^{1/2}\Phi)^\dagger\|}{1-\gamma}\sqrt{\psi_X^w}$
    Sample a $d$-dimensional vector $\eta \sim \mathcal{N}(0,\sigma_X^2 I)$
    Return $\hat{\theta}_X^w = \theta_X^w + \eta$
```

The variance of the noise in DP-LSW is proportional to the upper bound $R_{\max}/(1-\gamma)$ on the return from any state. This bound might be excessively pessimistic in some applications, leading to unnecessary large perturbation of the solution θ_X^w. Fortunately, it is possible to replace the term $R_{\max}/(1-\gamma)$ with any smaller upper bound $F_{\max}$ on the returns generated by the target MDP on any state. In practice this leads to more useful algorithms, but it is important to keep in mind that for the privacy guarantees to remain unaffected, one needs to assume that $F_{\max}$ is a publicly known quantity (i.e. it is not based on an estimate made from private data). These same considerations apply to the algorithm in the next section.

3.3.2 Algorithm DP-LSL

The second DP algorithm for policy evaluation we propose is also an output perturbation mechanism. It differs from DP-LSW in the way stability of the unperturbed solutions is promoted. In this case, we choose to optimize a regularized version of $J_X(\theta)$. In particular, we consider the objective function $J_X^\lambda(\theta)$ obtained by adding a ridge penalty to the least-squares loss from (3.1):

$$J_X^\lambda(\theta) = J_X(\theta) + \frac{\lambda}{2m}\|\theta\|_2^2 \ , \tag{3.8}$$

where $\lambda > 0$ is a regularization parameter. The introduction of the ridge penalty makes the objective function $J_X^\lambda(\theta)$ strongly convex, and thus ensures the existence of a unique solution $\theta_X^\lambda = \arg\min_\theta J_X^\lambda(\theta)$, which can be obtained in closed-form as:

$$\theta_X^\lambda = \left(\Phi^\top \Gamma_X \Phi + \frac{\lambda}{2m} I\right)^{-1} \Phi^\top \Gamma_X F_X \ . \tag{3.9}$$

Here Γ_X is defined as in Section 3.2.1.

We call DP-LSL the algorithm obtained by applying an output perturbation mechanism to the minimizer of $J_X^\lambda(\theta)$; the full pseudo-code is given in Algorithm 2. It receives as input the privacy parameters ε and δ, a dataset of trajectories X, the regression weights ρ, the feature representation Φ, a regularization parameter $\lambda > \|\Phi\|^2\|\rho\|_\infty$, and the MDP parameters $R_{\max}$ and γ. After computing the solution θ_X^λ to $\arg\min_\theta J_X^\lambda(\theta)$, the algorithm outputs $\hat\theta_X^\lambda = \theta_X^\lambda + \eta$, where η is a d-dimensional noise vector drawn from $\mathcal{N}(0, \sigma_X^2 I)$. The variance of η is obtained by solving a discrete optimization problem (different from the one in DP-LSW). Let $c_\lambda = \|\Phi\|\|\rho\|_\infty/\sqrt{2\lambda}$ and for $k \geq 0$, define $\varphi_X^\lambda(k)$ as:

$$\left(c_\lambda \sqrt{\sum_s \rho_s \min\{|X_s| + k, m\}} + \|\rho\|_2\right)^2 \ . \tag{3.10}$$

Then DP-LSL computes $\psi_X^\lambda = \max_{0\leq k\leq m} e^{-k\beta}\varphi_X^\lambda(k)$, which can be done in time $O(mN)$.

Algorithm 2 DP-LSL

```
Require: $X$, $\Phi$, $\gamma$, $R_{\max}$, $\rho$, $\lambda$, $\varepsilon$, $\delta$
    Compute $\theta_X^\lambda$                                   ▷ cf. (3.9)
    Let $\alpha \leftarrow \frac{5\sqrt{2\ln(2/\delta)}}{\varepsilon}$ and $\beta \leftarrow \frac{\varepsilon}{4(d+\ln(2/\delta))}$
    Let $\psi_X^\lambda \leftarrow \max_{0\leq k\leq m} e^{-k\beta}\varphi_X^\lambda(k)$            ▷ cf. (3.10)
    Let $\sigma_X \leftarrow \frac{2\alpha R_{\max}\|\Phi\|}{(1-\gamma)(\lambda-\|\Phi\|^2\|\rho\|_\infty)}\sqrt{\psi_X^\lambda}$
    Sample a $d$-dimensional vector $\eta \sim \mathcal{N}(0, \sigma_X^2 I)$
    Return $\hat\theta_X^\lambda = \theta_X^\lambda + \eta$
```

3.4 Privacy Analysis

This section provides a formal privacy analysis for DP-LSW and DP-LSL and shows that both algorithms are (ε, δ)-differentially private. In fact, for many applications (like the one we consider in this chapter) the global sensitivity is too large to provide useful mechanisms. Ideally one would like to add perturbations proportional to the potential changes around the input dataset X, as measured, for example by the *local sensitivity* $\mathrm{LS}_p(f,X) = \sup_{X' \simeq X} \|f(X) - f(X')\|_p$. Nissim et al. (2007) showed that approaches based on LS_p do not lead to differentially private algorithms, and then proposed an alternative framework for DP mechanisms with data-dependent perturbations based on the idea of *smoothed sensitivity*. This is the approach we use in this chapter. We use the smooth sensitivity framework of (Nissim et al., 2007, 2011), which provides tools for the design of DP mechanisms with data-dependent output perturbations.

3.4.1 Smoothed Gaussian Perturbation

We rely on the following lemma, which provides sufficient conditions for calibrating Gaussian output perturbation mechanisms with variance proportional to smooth upper bounds of the local sensitivity.

Lemma 3.1 (Nissim et al. (2011)). *Let A be an algorithm that on input X computes a vector $\mu_X \in \mathbb{R}^d$ deterministically and then outputs $Z_X \sim \mathcal{N}(\mu_X, \sigma_X^2 I)$, where σ_X^2 is a variance that depends on X. Let $\alpha = \alpha(\varepsilon,\delta) = 5\sqrt{2\ln(2/\delta)}/\varepsilon$ and $\beta = \beta(\varepsilon,\delta,d) = \varepsilon/(4d + 4\ln(2/\delta))$. Suppose ε and δ are such that the following are satisfied for every pair of neighbouring datasets $X \simeq X'$: (a) $\sigma_X \geq \alpha\|\mu_X - \mu_{X'}\|_2$, and (b) $|\ln(\sigma_X^2) - \ln(\sigma_{X'}^2)| \leq \beta$. Then A is (ε,δ)-differentially private.*

Condition (a) says we need variance at least proportional to the local sensitivity $\mathrm{LS}_2(f,X)$. Condition (b) asks that the variance does not change too fast between neighbouring datasets, by imposing the constraint $\sigma_X^2/\sigma_{X'}^2 \leq e^{\beta}$. This is precisely the spirit of the smoothed sensitivity principle: calibrate the noise to a smooth upper bound of the local sensitivity. A proof of Lemma 3.1 can be found in the pre-print Nissim et al. (2011). For the sake of completeness, we provide here an elementary proof (albeit with slightly worse constants). We acknowledge Lemma 3.1 is only available in pre-print form, and thus provide an elementary proof later in this Section for completeness. In particular, we are going to prove the following.

Lemma 3.2. *Let A be an algorithm that on input X computes a vector $\mu_X \in \mathbb{R}^d$ deterministically and then outputs $Z_X \sim \mathcal{N}(\mu_X, \sigma_X^2 I)$, where σ_X^2 is a variance that depends on X. Let $\alpha = \alpha(\varepsilon,\delta) = 15\sqrt{2\ln(4/\delta)}/\varepsilon$ and $\beta = \beta(\varepsilon,\delta,d) = (2\ln 2)\varepsilon/5(\sqrt{d} +$*

$\sqrt{2\ln(4/\delta)})^2$. *Suppose that* $\varepsilon \leq 5$, δ *and* d *are such* $\beta \leq \ln 2$, *and the following are satisfied for every pair of neighbouring datasets* $X \simeq X'$*:*

1. $\sigma_X \geq \alpha \|\mu_X - \mu_{X'}\|_2$,
2. $|\ln(\sigma_X^2) - \ln(\sigma_{X'}^2)| \leq \beta$.

Then A *is* (ε, δ)*-differentially private.*

We start with a simple characterization of (ε, δ)-differential privacy that will be useful for our proof.

Lemma 3.3. *Let* $A(X) = \theta_X \in \mathbb{R}^d$ *be the output of a randomized algorithm on input* X. *Write* $f_{\theta_X}(\theta)$ *for the probability density of the output of* A *on input* X. *Suppose that for every pair of neighbouring datasets* $X \simeq X'$ *there exists a measurable set* $\Theta_{X,X'} \subset \mathbb{R}^d$ *such that the following are satisfied:*

1. $\mathbb{P}[\theta_X \notin \Theta_{X,X'}] \leq \delta$;
2. *for all* $\theta \in \Theta_{X,X'}$ *we have* $f_{\theta_X}(\theta) \leq e^\varepsilon f_{\theta_{X'}}(\theta)$.

Then A *is* (ε, δ)*-differentially private.*

Proof. Fix a pair of neighbouring datasets $X \simeq X'$ and let $E \subseteq \mathbb{R}^d$ be any measurable set. Let $\Theta_{X,X'}$ be as in the statement and write $\Theta^c_{X,X'} = \mathbb{R}^d \setminus \Theta_{X,X'}$. Using the assumptions on $\Theta_{X,X'}$ we see that

$$\begin{aligned}
\mathbb{P}[\theta_X \in E] &= \mathbb{P}[\theta_X \in E \cap \Theta_{X,X'}] + \mathbb{P}[\theta_X \in E \cap \Theta^c_{X,X'}] \\
&\leq e^\varepsilon \mathbb{P}[\theta_{X'} \in E \cap \Theta_{X,X'}] + \delta \\
&\leq e^\varepsilon \mathbb{P}[\theta_{X'} \in E] + \delta \ .
\end{aligned}$$

□

Now we proceed with the proof of Lemma 3.2. Let $X \simeq X'$ be two neighbouring datasets and let us write $Z_1 = Z_X$ and $Z_2 = Z_{X'}$ for simplicity. Thus, for $i = 1, 2$ we have that $Z_i \sim \mathcal{N}(\mu_i, \sigma_i^2 I)$ are d-dimensional independent Gaussian random variables whose means and variances satisfy the assumptions of Lemma 3.2 for some $\varepsilon, \delta > 0$.

The density function of Z_i is denoted by $f_{Z_i}(z)$. In order to be able to apply Lemma 3.3 we want to show that the privacy loss between Z_1 and Z_2 defined as

$$L(z) = \ln \frac{f_{Z_1}(z)}{f_{Z_2}(z)} \tag{3.11}$$

is bounded by ε for all $z \in \Omega$, where $\Omega \subset \mathbb{R}^d$ is an event with probability at least $1-\delta$ under Z_1.

We can start by identifying a candidate Ω. Since Ω has to have high probability w.r.t. Z_1, it should contain μ_1 because a ball around the mean is the event with the highest probability under a spherical Gaussian distribution (among those with the same Lebesgue measure). For technical reasons, instead of a ball we will take a slightly more complicated region, which for now we will parametrize by two quantities $a, b > 0$. The definition of this region will depend on the difference of means $\Delta = \mu_2 - \mu_1$:

$$\Omega = \Omega_a \cap \Omega_b = \{z + \mu_1 \in \mathbb{R}^d \mid |\langle z, \Delta\rangle| \leq a\} \cap \{z + \mu_1 \in \mathbb{R}^d \mid \|z\| \leq b\} \ . \tag{3.12}$$

We need to choose a and b such that the probability $\mathbb{P}[Z_1 \notin \Omega] \leq \delta$, and for that we shall combine two different tail bounds. On the one hand, note that $Z = \langle Z_1 - \mu_1, \Delta\rangle / (\sigma_1 \|\Delta\|) \sim \mathcal{N}(0,1)$ is a one dimensional standard Gaussian random variable and recall that for any $t \geq 0$:

$$\mathbb{P}[|Z| > t] \leq 2e^{-t^2/2} \ . \tag{3.13}$$

On the other hand, $X = \|Z_1 - \mu_1\|^2/\sigma_1^2 \sim \chi_d^2$ follows a chi-squared distribution with d degrees of freedom, for which is known Laurent and Massart (2000) that for all $t \geq 0$:

$$\mathbb{P}[X > d + 2\sqrt{dt} + 2t] \leq e^{-t} \ . \tag{3.14}$$

To make our choices for a and b we can take them such that $\mathbb{P}[Z_1 \notin \Omega_a], \mathbb{P}[Z_1 \notin \Omega_b] \leq \delta/2$, since then by a union bound we will get

$$\mathbb{P}[Z_1 \notin \Omega] \leq \mathbb{P}[Z_1 \notin \Omega_A] + \mathbb{P}[Z_1 \notin \Omega_B] \leq \delta \ . \tag{3.15}$$

Since Z satisfies $|Z| \leq \sqrt{2\ln(4/\delta)}$ with probability at least $1-\delta/2$, we can take

$$a = \sigma_1 \|\Delta\| \sqrt{2 \ln \frac{4}{\delta}} = \sigma_1 \|\Delta\| C_\delta \ . \tag{3.16}$$

For X we have that $d + 2\sqrt{d \ln(2/\delta)} + 2\ln(2/\delta) \leq d + 2\sqrt{2d \ln(2/\delta)} + 2\ln(2/\delta) = (\sqrt{d} + \sqrt{2\ln(2/\delta)})^2$. Hence, we choose

$$b = \sigma_1(\sqrt{d} + \sqrt{2\ln(2/\delta)}) = \sigma_1 D_\delta \ . \tag{3.17}$$

Fixing this choice of Ω, we now proceed to see under what conditions on σ_1 and σ_2 we can get $L(z) \leq \varepsilon$ for all $z \in \Omega$. We start by expanding the definition of $L(z)$ to get

$$L(z) = \frac{d}{2} \ln \frac{\sigma_2^2}{\sigma_1^2} + \frac{\|\mu_2 - z\|^2}{2\sigma_2^2} - \frac{\|\mu_1 - z\|^2}{2\sigma_1^2} \ . \tag{3.18}$$

The easiest thing to do is to separate this quantity into several parts and insist on each part being at most a fraction of ε. To simplify calculations we will just require that each part is at most $\epsilon = \varepsilon/5$. This reasoning applied to the first term shows that we must satisfy

$$\frac{\sigma_2^2}{\sigma_1^2} \leq e^{2\epsilon/d} \ . \tag{3.19}$$

Note that this becomes more restrictive as $\epsilon \approx 0$ or $d \to \infty$, in which case we have $e^{\epsilon/d} \approx 1$.

Next we look at the second part and write $z = z' + \mu_1$ because this is the form of the vectors in Ω. With some algebra we get:

$$\frac{\|\mu_2 - (z' + \mu_1)\|^2}{2\sigma_2^2} - \frac{\|\mu_1 - (z' + \mu_1)\|^2}{2\sigma_1^2} = \frac{\|\Delta\|^2 + \|z'\|^2 - 2\langle z', \Delta\rangle}{2\sigma_2^2} - \frac{\|z'\|^2}{2\sigma_1^2} \ . \tag{3.20}$$

To further decompose this quantity we write $z' \in \mathbb{R}^d$ as $z' = z_p + z_o$, where $z_p = \Delta \langle z', \Delta\rangle / \|\Delta\|^2$ is the orthogonal projection of z onto the line spanned by the vector Δ, and z_o is the corresponding orthogonal complement. Pythagora's Theorem implies $\|z'\|^2 = \|z_p\|^2 + \|z_o\|^2$, and the RHS in the above expression is equal to

$$\frac{\|\Delta\|^2}{2\sigma_2^2} - \frac{\langle z', \Delta\rangle}{\sigma_2^2} + \frac{|\langle z', \Delta\rangle|^2}{2\|\Delta\|^2}\left(\frac{1}{\sigma_2^2} - \frac{1}{\sigma_1^2}\right) + \frac{\|z_o\|^2}{2}\left(\frac{1}{\sigma_2^2} - \frac{1}{\sigma_1^2}\right) \ . \tag{3.21}$$

Now note that the last two terms can be upper bounded by zero if $\sigma_1 \leq \sigma_2$, but need to be taken into account otherwise. Furthermore, if it were the case that $\sigma_1 \gg \sigma_2 \approx 0$, then these terms could grow unboundedly. Thus we shall require that a bound of the form

$$\frac{\sigma_1^2}{\sigma_2^2} \leq \gamma \ , \tag{3.22}$$

holds for some $\gamma \geq 1$ to be specified later. Nonetheless, we observe that under this assumption

$$\frac{1}{\sigma_2^2} - \frac{1}{\sigma_1^2} \leq \frac{\gamma - 1}{\sigma_1^2} \ . \tag{3.23}$$

Furthermore, $z \in \Omega$ implies $\|z_o\|^2 \leq \|z'\|^2 = \|z - \mu_1\| \leq b^2$ and $|\langle z', \Delta \rangle|^2 = |\langle z - \mu_1, \Delta \rangle|^2 \leq a^2$. Thus we see that

$$\frac{|\langle z', \Delta \rangle|^2}{2\|\Delta\|^2} \left(\frac{1}{\sigma_2^2} - \frac{1}{\sigma_1^2} \right) \leq \frac{C_\delta^2(\gamma - 1)}{2} \ , \tag{3.24}$$

and

$$\frac{\|z_o\|^2}{2} \left(\frac{1}{\sigma_2^2} - \frac{1}{\sigma_1^2} \right) \leq \frac{D_\delta^2(\gamma - 1)}{2} \ . \tag{3.25}$$

By requiring that each of these bounds is at most ϵ we obtain the following constraint for γ:

$$\gamma \leq 1 + \frac{2\epsilon}{\max\{C_\delta^2, D_\delta^2\}} \ , \tag{3.26}$$

which can be satisfied by taking, for example:

$$\gamma = 1 + \frac{2\epsilon}{\left(\sqrt{d} + \sqrt{2\ln(4/\delta)}\right)^2} \ . \tag{3.27}$$

Note that for fixed δ, small ϵ and/or large d this choice of γ will make (3.22) behave much like the bound (3.19) we assumed above for σ_2^2/σ_1^2. In fact, using that $1 + x \geq e^{x \ln 2}$ for all $0 \leq x \leq 1$ we see that (3.22) can be satisfied if $2\epsilon/(\sqrt{d} + \sqrt{2\ln(4/\delta)})^2 \leq 1$ and

$$\frac{\sigma_1^2}{\sigma_2^2} \leq \exp\left(\frac{(2\ln 2)\epsilon}{\left(\sqrt{d} + \sqrt{2\ln(4/\delta)}\right)^2} \right) \ . \tag{3.28}$$

From here it is immediate to see that if the second condition $|\ln(\sigma_1^2) - \ln(\sigma_2^2)| \leq \beta$ in Lemma 3.2 is satisfied, then (3.19) and (3.28) are both satisfied.

The missing ingredient to show that $L(z) \leq \varepsilon$ for all $z \in \Omega$ is an absolute lower bound on σ_1. This will follow from bounding the remaining terms in $L(z)$ as follows:

$$\begin{aligned} \frac{\|\Delta\|^2}{2\sigma_2^2} - \frac{\langle z', \Delta \rangle}{\sigma_2^2} &\leq \frac{\|\Delta\|^2 + 2\sigma_1 \|\Delta\| C_\delta}{2\sigma_2^2} && (3.29) \\ &\leq \frac{\gamma}{2} \frac{\|\Delta\|^2 + 2\sigma_1 \|\Delta\| C_\delta}{\sigma_1^2} && (3.30) \\ &\leq \frac{3}{2} \frac{\|\Delta\|^2 + 2\sigma_1 \|\Delta\| C_\delta}{\sigma_1^2} && (3.31) \\ &= \frac{3\|\Delta\|^2}{2\sigma_1^2} + \frac{3\|\Delta\| C_\delta}{\sigma_1} \ , && (3.32) \end{aligned}$$

where we used that $\epsilon \leq 1$ implies $\gamma \leq 3$. If we require each of these two terms to be at most ϵ, we obtain the constraint:

$$\sigma_1 \geq \|\Delta\| \max \left\{ \sqrt{\frac{3}{2\epsilon}}, \frac{3C_\delta}{\epsilon} \right\} = \frac{3\|\Delta\| C_\delta}{\epsilon} \ . \tag{3.33}$$

To conclude the proof just note that the above bound can be rewritten as $\sigma_1 \geq \alpha \|\Delta\|$, which is precisely the first condition in Lemma 3.2.

3.4.2 Privacy Analysis of DP-LSW

We start by providing an upper bound on the norm $\|\theta_X^w - \theta_{X'}^w\|_2$ for any two neighbouring datasets $X \simeq X'$. Using (3.6) it is immediate that:

$$\|\theta_X^w - \theta_{X'}^w\|_2 \leq \|(\Gamma^{1/2}\Phi)^\dagger\| \|F_X - F_{X'}\|_{2,\Gamma} \ . \tag{3.34}$$

Thus, we need to bound $\|F_X - F_{X'}\|_{2,\Gamma}$.

Lemma 3.4. *Let $X \simeq X'$ be two neighbouring datasets of m trajectories with $X = (x_1, \ldots, x_{m-1}, x)$ and $X' = (x_1, \ldots, x_{m-1}, x')$. Let $X^\circ = (x_1, \ldots, x_{m-1})$. Let $\mathcal{S}_x$ (resp. $\mathcal{S}_{x'}$) denote the set of states visited by x (resp. x'). Then we have*

$$\|F_X - F_{X'}\|_{2,\Gamma} \leq \frac{R_{\max}}{1-\gamma} \sqrt{\sum_{s \in \mathcal{S}_x \cup \mathcal{S}_{x'}} \frac{w_s}{(|X_s^\circ| + 1)^2}} \ .$$

Proof. We start by noting that if $s \in \mathcal{S} \setminus (\mathcal{S}_x \cup \mathcal{S}_{x'})$, then $F_{X,s} = F_{X',s}$. In the case $s \in \mathcal{S}_x \cup \mathcal{S}_{x'}$ we can write $F_{X,s} = (|X_s^\circ| F_{X^\circ,s} + F_{x,s}) / (|X_s^\circ| + 1)$. Using a symmetric expression for $F_{X',s}$ we see that in this case

$$|F_{X,s} - F_{X',s}| = \frac{1}{|X_s^\circ| + 1} |F_{x,s} - F_{x',s}| \leq \frac{1}{|X_s^\circ| + 1} \max\{F_{x,s}, F_{x',s}\} \leq \frac{1}{|X_s^\circ| + 1} \frac{R_{\max}}{1 - \gamma} ,$$

where we used that $0 \leq F_{x,s} \leq R_{\max}/(1-\gamma)$ for all s and x. When $s \in \mathcal{S}_x \setminus \mathcal{S}_{x'}$ we can use the same expression as before for $F_{X,s}$ and write $F_{X',s} = F_{X^\circ,s}$. A similar argument as in the previous case then yields

$$|F_{X,s} - F_{X',s}| = \frac{1}{|X_s^\circ| + 1} |F_{x,s} - F_{X^\circ,s}| \leq \frac{1}{|X_s^\circ| + 1} \frac{R_{\max}}{1 - \gamma} .$$

Note the same bound also holds for the case $s \in \mathcal{S}_{x'} \setminus \mathcal{S}_x$. Finally, since we have seen that the same bound holds for all $s \in \mathcal{S}_x \cup \mathcal{S}_{x'}$, we obtain

$$\sum_{s \in \mathcal{S}} w_s (F_{X,s} - F_{X',s})^2 \leq \frac{R_{\max}^2}{(1-\gamma)^2} \sum_{s \in \mathcal{S}_x \cup \mathcal{S}_{x'}} \frac{w_s}{(|X_s^\circ| + 1)^2} ,$$

which yields the desired bound. □

Since the condition in Lemma 3.1 needs to hold for any dataset X' neighbouring X, we take the supremum of the bound above over all neighbours., which yields the following corollary.

Corollary 3.5. *If X is a dataset of trajectories, then the following holds for every neighbouring dataset $X' \simeq X$:*

$$\|F_X - F_{X'}\|_{2,\Gamma} \leq \frac{R_{\max}}{1-\gamma} \sqrt{\sum_{s \in \mathcal{S}} \frac{w_s}{\max\{|X_s|, 1\}^2}} .$$

Proof. Using the notation from Lemma 3.4 we observe that $|X_s| = |X_s^\circ| + 1$ if $s \in \mathcal{S}_x$, and $|X_s| = |X_s^\circ|$ if $s \notin \mathcal{S}_x$. Therefore, the following holds for any trajectories x, x':

$$\begin{aligned}\sum_{s \in \mathcal{S}_x \cup \mathcal{S}_{x'}} \frac{w_s}{(|X_s^\circ| + 1)^2} \leq \sum_{s \in \mathcal{S}} \frac{w_s}{(|X_s^\circ| + 1)^2} &= \sum_{s \in \mathcal{S}_x} \frac{w_s}{|X_s|^2} + \sum_{s \in \mathcal{S} \setminus \mathcal{S}_x} \frac{w_s}{(|X_s| + 1)^2} \\ &\leq \sum_{s \in \mathcal{S}_X} \frac{w_s}{|X_s|^2} + \sum_{s \in \mathcal{S} \setminus \mathcal{S}_X} w_s ,\end{aligned}$$

where $\mathcal{S}_X$ denotes the set of states visited by at least one trajectory from X. Since $s \notin \mathcal{S}_X$ implies $|X_s| = 0$, we can plug this bound into the result of Lemma 3.4 as follows:

$$\|F_X - F_{X'}\|_{2,\Gamma} \leq \frac{R_{\max}}{1-\gamma}\sqrt{\sum_{s\in\mathcal{S}_X}\frac{w_s}{|X_s|^2} + \sum_{s\in\mathcal{S}\setminus\mathcal{S}_X} w_s} = \frac{R_{\max}}{1-\gamma}\sqrt{\sum_{s\in\mathcal{S}}\frac{w_s}{\max\{|X_s|,1\}^2}} \enspace .$$

□

Using this result we see that in order to satisfy item (a) of Lemma 3.1 we can choose a noise variance satisfying:

$$\sigma_X \geq \frac{\alpha R_{\max}\|(\Gamma^{1/2}\Phi)^\dagger\|}{1-\gamma}\sqrt{\sum_{s\in\mathcal{S}}\frac{w_s}{\max\{|X_s|,1\}^2}} \enspace , \tag{3.35}$$

where only the last multiplicative term depends on the dataset X, and the rest can be regarded as a constant that depends on parameters of the problem which are either public or chosen by the user, and will not change for a neighbouring dataset X'. Thus, we are left with a lower bound expressible as $\sigma_X \geq C\sqrt{\varphi_X^w}$, where $\varphi_X^w = \sum_s (w_s/\max\{|X_s|,1\}^2)$ only depends on the dataset X through its *signature* $\langle X\rangle \in \mathbb{N}^{\mathcal{S}}$ given by the number of times each state appears in the trajectories of X: $\langle X\rangle(s) = |X_s|$. Accordingly, we write $\varphi_X^w = \varphi^w(\langle X\rangle)$, where $\varphi^w : \mathbb{N}^{\mathcal{S}} \to \mathbb{R}$ is the function

$$\varphi^w(v) = \sum_s \frac{w_s}{\max\{v_s,1\}^2} \enspace . \tag{3.36}$$

The signatures of two neighbouring datasets $X \simeq X'$ satisfy $\|\langle X\rangle - \langle X'\rangle\|_\infty \leq 1$ because replacing a single trajectory can only change by one the number of first visits to any particular state. Thus, assuming we have a function $\psi : \mathbb{N}^{\mathcal{S}} \to \mathbb{R}$ satisfying $\psi^w(v) \geq \varphi^w(v)$ and $|\ln(\psi^w(v)) - \ln(\psi^w(v'))| \leq \beta$ for all $v, v' \in \mathbb{N}^{\mathcal{S}}$ with $\|v - v'\|_\infty \leq 1$, we can take $\sigma_X = C\sqrt{\psi^w(\langle X\rangle)}$. This variance clearly satisfies the conditions of Lemma 3.1 since

$$|\ln(\sigma_X^2) - \ln(\sigma_{X'}^2)| = |\ln(\psi^w(\langle X\rangle)) - \ln(\psi^w(\langle X'\rangle))| \leq \beta \enspace .$$

The function ψ^w is known as a *β-smooth upper bound* of φ^w, and the following result provides a tool for constructing such functions.

Lemma 3.6 (Nissim et al. (2007))**.** *Let $\varphi : \mathbb{N}^{\mathcal{S}} \to \mathbb{R}$. For any $k \geq 0$ let $\varphi_k(v) = \max_{\|v-v'\|_\infty\leq k}\varphi(v')$. Given $\beta > 0$, the smallest β-smooth upper bound of φ is the* *function*

$$\psi(v) = \sup_{k \geq 0} \left(e^{-k\beta} \varphi_k(v) \right) \quad . \tag{3.37}$$

For some functions φ, the upper bound ψ can be hard to compute or even approximate (Nissim et al., 2007). Fortunately, in our case a simple inspection of (3.36) reveals that $\varphi_k^w(v)$ is easy to compute. In particular, the following lemma implies that $\psi^w(v)$ can be obtained in time $O(N\|v\|_\infty)$.

Lemma 3.7. *The following holds for every $v \in \mathbb{N}^{\mathcal{S}}$:*

$$\varphi_k^w(v) = \sum_{s \in \mathcal{S}} \frac{w_s}{\max\{v_s - k, 1\}^2} \quad .$$

Furthermore, for every $k \geq \|v\|_\infty - 1$ we have $\varphi_k^w(v) = \sum_s w_s$.

Proof. Recall that $\varphi_k^w(v) = \max_{\|v' - v\|_\infty \leq k} \varphi^w(v')$ with $\varphi^w(v) = \sum_s w_s / \max\{v_s, 1\}^2$ and observe the result follows immediately because

$$\varphi_k^w(v) = \sum_{s \in \mathcal{S}} \frac{w_s}{\min_{-k \leq l \leq k} \max\{v_s + l, 1\}^2} = \sum_{s \in \mathcal{S}} \frac{w_s}{\max\{v_s - k, 1\}^2} \quad . \qquad \square$$

Combining the last two lemmas, we see that the quantity ψ_X^w computed in DP-LSW is in fact a β-smooth upper bound to φ_X^w. Because the variance σ_X used in DP-LSW can be obtained by plugging this upper bound into (3.35), the two conditions of Lemma 3.1 are satisfied. This completes the proof of the main result of this section:

Theorem 3.8. *Algorithm DP-LSW is (ε, δ)-differentially private.*

Before proceeding to the next privacy analysis, note that Corollary 3.5 is the reason why a mechanism with output perturbations proportional to the global sensitivity is not sufficient in this case. The bound there says that if in the worst case we can find datasets of an arbitrary size m where some states are visited few (or zero) times, then the global sensitivity will not vanish as $m \to \infty$. Hence, the utility of such algorithm would not improve with the size of the dataset. The smoothed sensitivity approach works around this problem by adding large noise to these datasets, but adding much less noise to datasets where each state appears a sufficient number of times. Corollary 3.5 also provides the basis for efficiently computing smooth upper bounds to the local sensitivity. In principle, condition (b) in Lemma 3.1 refers to any dataset neighbouring X, of which there are uncountably many because we consider real rewards. Bounding the local sensitivity in terms of the signature reduces this to finitely many "classes" of neighbours, and the form of the bound in Corollary 3.5 makes it possible to apply Lemma 3.6 efficiently.

3.4.3 Privacy Analysis of DP-LSL

The proof that DP-LSL is differentially private follows the same strategy as for DP-LSW. We start with a lemma that bounds the local sensitivity of θ_X^λ for pairs of neighbouring datasets $X \simeq X'$. We use the notation $\mathbb{I}_{s\in x}$ for an indicator variable that is equal to one when state s is visited within trajectory x.

Lemma 3.9. *Let $X \simeq X'$ be two neighbouring datasets of m trajectories with $X = (x_1, \dots, x_{m-1}, x)$ and $X' = (x_1, \dots, x_{m-1}, x')$. Let $F_x \in \mathbb{R}^{\mathcal{S}}$ (resp. $F_{x'} \in \mathbb{R}^{\mathcal{S}}$) be the vector given by $F_x(s) = F_{x,s}$ (resp. $F_{x'}(s) = F_{x',s}$). Define the diagonal matrices $\Gamma_\rho, \Delta_{x,x'} \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}$ given by $\Gamma_\rho(s,s) = \rho_s$ and $\Delta_{x,x'}(s,s) = \mathbb{I}_{s\in x} - \mathbb{I}_{s\in x'}$. If the regularization parameter satisfies $\lambda > \|\Phi^\top \Delta_{x,x'}\Gamma_\rho\Phi\|$, then the following holds:*

$$\frac{\|\theta_X^\lambda - \theta_{X'}^\lambda\|_2}{2} \le \frac{\left\| \left(\Delta_{x,x'}\Phi\theta_X^\lambda - F_x + F_{x'}\right)^\top \Gamma_\rho \Phi \right\|_2}{\lambda - \|\Phi^\top \Delta_{x,x'}\Gamma_\rho\Phi\|} \ . \tag{3.38}$$

Proof. In order to simplify our notation we write $\bar{\theta} = \theta_X^\lambda$ and $\bar{\theta}' = \theta_{X'}^\lambda$ for the rest of the proof. Given a trajectory x and a vector $\theta \in \mathbb{R}^d$ we shall also write $\ell(x,\theta) = \sum_{s\in\mathcal{S}_x} \rho_s (F_{x,s} - \phi_s^\top\theta)^2$ so that $J_X(\theta) = \frac{1}{m}\sum_{i=1}^m \ell(x_i,\theta)$. Now we proceed with the proof.

Let us start by noting that because $J_X^\lambda(\theta)$ is λ/m-strongly convex, we have $J_X^\lambda(\theta_1) - J_X^\lambda(\theta_2) \ge \langle \nabla J_X^\lambda(\theta_2), \theta_1 - \theta_2\rangle + \frac{\lambda}{2m}\|\theta_1 - \theta_2\|_2^2$ for any $\theta_1, \theta_2 \in \mathbb{R}^d$. Thus, using that optimality implies $\nabla J_X^\lambda(\bar{\theta}) = \nabla J_{X'}^\lambda(\bar{\theta}') = 0$, we get

$$\begin{aligned}
\frac{\lambda}{m}\|\bar{\theta} - \bar{\theta}'\|_2^2 &\le J_X^\lambda(\bar{\theta}') - J_X^\lambda(\bar{\theta}) + J_{X'}^\lambda(\bar{\theta}) - J_{X'}^\lambda(\bar{\theta}') \\
&= J_X(\bar{\theta}') - J_X(\bar{\theta}) + J_{X'}(\bar{\theta}) - J_{X'}(\bar{\theta}') \\
&= \frac{1}{m}\left(\ell(x,\bar{\theta}') - \ell(x,\bar{\theta}) + \ell(x',\bar{\theta}) - \ell(x',\bar{\theta}')\right) \ ,
\end{aligned}$$

where the equalities follows from definitions of X, X', J_X^λ and J_X. If we now expand the definition of $\ell(x,\theta)$ we see that

$$\begin{aligned}
\ell(x,\bar{\theta}') - \ell(x,\bar{\theta}) &= \sum_{s\in\mathcal{S}_x} \rho_s \left((\phi_s^\top \bar{\theta}')^2 - (\phi_s^\top \bar{\theta})^2 - 2F_{x,s}\phi_s^\top(\bar{\theta}' - \bar{\theta}) \right) , \\
\ell(x',\bar{\theta}) - \ell(x',\bar{\theta}') &= \sum_{s\in\mathcal{S}_{x'}} \rho_s \left((\phi_s^\top \bar{\theta})^2 - (\phi_s^\top \bar{\theta}')^2 - 2F_{x',s}\phi_s^\top(\bar{\theta} - \bar{\theta}') \right) .
\end{aligned}$$

Using the identity $(\phi_s^\top \bar{\theta}')^2 - (\phi_s^\top \bar{\theta})^2 = (\bar{\theta}' + \bar{\theta})^\top \phi_s \phi_s^\top (\bar{\theta}' - \bar{\theta})$, we rewrite $\ell(x,\bar{\theta}') - \ell(x,\bar{\theta}) + \ell(x',\bar{\theta}) - \ell(x',\bar{\theta}')$ as

$$\sum_{s\in\mathcal{S}} \rho_s \left[(\mathbb{I}_{s\in x} - \mathbb{I}_{s\in x'})(\bar{\theta}' + \bar{\theta})^\top \phi_s \phi_s^\top - 2(F_{x,s} - F_{x',s})\phi_s^\top \right] (\bar{\theta}' - \bar{\theta}) , \tag{3.39}$$

where we implicitly used that $F_{x,s} = 0$ whenever $s \notin x$. Finally, using the definitions in the statement we can rearrange the above expression to show that

$$\begin{aligned}
&\frac{\lambda}{m}\|\bar{\theta} - \bar{\theta}'\|_2^2 \leq \frac{1}{m}\left((\bar{\theta}' + \bar{\theta})^\top \Phi^\top \Delta_{x,x'} - 2(F_x - F_{x'})^\top \right) \Gamma_\rho \Phi (\bar{\theta}' - \bar{\theta}) \\
&= \frac{2}{m}\left(\bar{\theta}^\top \Phi^\top \Delta_{x,x'} - (F_x - F_{x'})^\top \right) \Gamma_\rho \Phi(\bar{\theta}' - \bar{\theta}) + \frac{1}{m}(\bar{\theta}' - \bar{\theta})^\top \Phi^\top \Delta_{x,x'} \Gamma_\rho \Phi (\bar{\theta}' - \bar{\theta}) \\
&\leq \frac{2}{m}\| \left(\bar{\theta}^\top \Phi^\top \Delta_{x,x'} - (F_x - F_{x'})^\top \right) \Gamma_\rho \Phi \|_2 \|\bar{\theta}' - \bar{\theta}\|_2 + \frac{1}{m}\|\Phi^\top \Delta_{x,x'} \Gamma_\rho \Phi\| \|\bar{\theta}' - \bar{\theta}\|_2^2 ,
\end{aligned}$$

where we used the Cauchy–Schwartz inequality and the definition of operator norm. The result now follows by solving for $\|\bar{\theta} - \bar{\theta}'\|_2$ in the above inequality. □

As before, we need to consider the supremum of the bound over all possible neighbours X' of X. In particular, we would like to get a bound whose only dependence on the dataset X is through the signature $\langle X \rangle$. This is the purpose of the following corollary:

Corollary 3.10. *Let X be a dataset of trajectories and suppose $\lambda > \|\Phi\|^2 \|\rho\|_\infty$. Then the following holds for every neighbouring dataset $X' \simeq X$:*

$$\|\theta_X^\lambda - \theta_{X'}^\lambda\|_2 \leq \frac{2R_{\max}\|\Phi\|}{(1-\gamma)(\lambda - \|\Phi\|^2\|\rho\|_\infty)} \sqrt{\varphi_X^\lambda} ,$$

where

$$\varphi_X^\lambda = \left(\frac{\|\Phi\| \|\rho\|_\infty}{\sqrt{2\lambda}} \sqrt{\sum_{s\in\mathcal{S}} \rho_s |X_s|} + \|\rho\|_2 \right)^2 .$$

Proof. We start by noting that $\|\Delta_{x,x'}\| \leq 1$ and $\|\Gamma_\rho\| = \|\rho\|_\infty$, hence submultiplicativity of matrix operator norms yields $\|\Phi^\top \Delta_{x,x'} \Gamma_\rho \Phi\| \leq \|\Phi\|^2 \|\rho\|_\infty$. On the other hand, for the numerator in (3.38) we have

$$\left\| \left(\Delta_{x,x'} \Phi \theta_X^\lambda - F_x + F_{x'} \right)^\top \Gamma_\rho \Phi \right\|_2 \leq \left(\|\theta_X^\lambda\|_2 \|\Phi\| \|\rho\|_\infty + \|(F_x - F_{x'})^\top \Gamma_\rho\|_2 \right) \|\Phi\| \ . \tag{3.40}$$

Bounding the individual entries in F_x and $F_{x'}$ by $R_{\max}/(1-\gamma)$ we get $\|(F_x - F_{x'})^\top \Gamma_\rho\|_2 \leq R_{\max}\|\rho\|_2/(1-\gamma)$. The last step is to bound the norm $\|\theta_X^\lambda\|_2$, for which we use the closed-form solution to $\arg\min_\theta J_X^\lambda(\theta)$ given in this chapter and write:

$$\begin{aligned} \|\theta_X^\lambda\|_2 &\leq \left\| (\Phi^\top \Gamma_X \Phi + \frac{\lambda}{2m} I)^{-1} \Phi^\top \Gamma_X^{1/2} \right\| \|F_X\|_{2,\Gamma_X} \\ &\leq \left\| (\Phi^\top \Gamma_X \Phi + \frac{\lambda}{2m} I)^{-1} \Phi^\top \Gamma_X^{1/2} \right\| \left(\frac{R_{\max}}{1-\gamma} \sqrt{\sum_{s \in \mathcal{S}} \frac{\rho_s |X_s|}{m}} \right) \ . \end{aligned}$$

To bound the last remaining norm let use write $U\Sigma V^\top$ for the SVD of $\Gamma_X^{1/2}\Phi$, where $V \in \mathbb{R}^{d\times d}$ with $V^\top V = VV^\top = I$. With this we can write:

$$(\Phi^\top \Gamma_X \Phi + \frac{\lambda}{2m} I)^{-1} \Phi^\top \Gamma_X^{1/2} = V \left(\Sigma^2 + \frac{\lambda}{2m} I \right)^{-1} \Sigma U^\top \ . \tag{3.41}$$

Now we use that $\|U\| = \|V\| = 1$ and $x/(x^2+a) \leq 1/(2\sqrt{a})$ for any $x \geq 0$ to get $\|V(\Sigma^2 + (\lambda/2m)I)^{-1}\Sigma U^\top\| \leq \sqrt{m/2\lambda}$. Thus we get a bound for $\|\theta_X^\lambda\|_2$ that when plugged into (3.40) yields the desired result. □

By the same reasoning of Section 3.4.2, as long as the regularization parameter is larger than $\|\Phi\|^2\|\rho\|_\infty$, a differentially private algorithm can be obtained by adding to θ_X^λ a Gaussian perturbation with a variance satisfying

$$\sigma_X \geq \frac{2\alpha R_{\max}\|\Phi\|}{(1-\gamma)(\lambda - \|\Phi\|^2\|\rho\|_\infty)} \sqrt{\varphi_X^\lambda}$$

and the second condition of Lemma 3.1. This second requirement can be achieved by computing a β-smooth upper bound of the function $\varphi^\lambda : \mathbb{N}^{\mathcal{S}} \to \mathbb{R}$ given by

$$\varphi^\lambda(v) = \left(\frac{\|\Phi\|\|\rho\|_\infty}{\sqrt{2\lambda}} \sqrt{\sum_{s\in\mathcal{S}} \rho_s \max\{v_s, m\}} + \|\rho\|_2 \right)^2 \ .$$

When going from φ_X^λ to $\varphi^\lambda(v)$ we substituted $|X_s|$ by $\max\{v_s, m\}$ to reflect the fact that any state cannot be visited by more than m trajectories in a dataset X of size m. It turns out that in this case the function $\varphi_k^\lambda(v) = \max_{\|v-v'\|_\infty \leq k} \varphi^\lambda(v')$ arising in Lemma 3.6 is also easy to compute.

Lemma 3.11. *For every $v \in \mathbb{N}^{\mathcal{S}}$, $\varphi_k^\lambda(v)$ is equal to:*

$$\left(\frac{\|\Phi\| \|\rho\|_\infty}{\sqrt{2\lambda}} \sqrt{\sum_{s \in \mathcal{S}} \rho_s \max\{v_s + k, m\}} + \|\rho\|_2 \right)^2 .$$

Furthermore, for every $k \geq m - \min_s v_s$ we have $\varphi_k^\lambda(v) = \left(\frac{\|\Phi\| \|\rho\|_\infty \sqrt{m}}{\sqrt{2\lambda}} \sqrt{\sum_{s \in \mathcal{S}} \rho_s} + \|\rho\|_2 \right)^2$.

Proof. The proof is similar to that of Lemma 3.7. □

Finally, in view of Lemma 3.6, Corollary 3.10, and Lemma 3.11, the variance of the noise perturbation in DP-LSL satisfies the conditions of Lemma 3.1, so we have proved the following.

Theorem 3.12. *Algorithm DP-LSL is (ε, δ)-differentially private.*

3.5 Utility Analysis

Because the promise of differential privacy has to hold for any possible pair of neighbouring datasets $X \simeq X'$, the analysis in previous section does not assume any generative model for the input dataset X. However, in practical applications we expect $X = (x_1, \ldots, x_m)$ to contain multiple trajectories sampled from the same policy on the same MDP. The purpose of this section is to show that when the trajectories x_i are i.i.d. the utility of our differentially private algorithms increases as $m \to \infty$. In other words, when the input dataset grows, the amount of noise added by our algorithms decreases, thus leading to more accurate estimates of the value function. This matches the intuition that when outputting a fixed number of parameters, using data from more users to estimate these parameters leads to a smaller individual contributions from each user, and makes the privacy constraint easier to satisfy.

To measure the utility of our DP algorithms we shall bound the difference in empirical risk between the private and non-private parameters learned from a given dataset. That is, we want to show that the quantity $\mathbb{E}_{X,\eta}\left[J_X^\bullet(\hat{\theta}_X^\bullet) - J_X^\bullet(\theta_X^\bullet)\right]$ vanishes as $|X| = m \to \infty$, for both $\bullet = w$ and $\bullet = \lambda$.

3.6 Utility Analysis of DP-LSW

The goal of this section is to show that as the size m of the dataset X grows, the differentially private solution θ_X^w provided by algorithm DP-LSW is not much worse than the one obtained by directly minimizing $J_X^w(\theta)$. In other words, for large datasets the noise introduced by the privacy constraint is negligible. We do so by proving a $O(1/m^2)$ bound for the expected empirical excess risk given by $\mathbb{E}_{X,\eta}\left[J_X^w(\hat{\theta}_X^w) - J_X^w(\theta_X^w)\right]$. Our analysis starts with a lemma that leverages the law of total expectation in order to reduce the bound to a quantity that only depends on $\mathbb{E}_X\left[\sigma_X^2\right]$.

Lemma 3.13.

$$\mathbb{E}_{X,\eta}\left[J_X^w(\hat{\theta}_X^w) - J_X^w(\theta_X^w)\right] = \|\Gamma^{1/2}\Phi\|_F^2 \mathbb{E}_X\left[\sigma_X^2\right] \ . \tag{3.42}$$

Proof. By the law of total expectation it is enough to show that

$$\mathbb{E}_\eta\left[J_X^w(\hat{\theta}_X^w) - J_X^w(\theta_X^w)|X\right] = \sigma_X^2\|\Gamma^{1/2}\Phi\|_F^2 \ . \tag{3.43}$$

Let X be an arbitrary dataset. Expanding the definition of $J_X^w(\theta)$ we have that for any $\theta \in \mathbb{R}^d$

$$J_X^w(\theta) = F_X^\top \Gamma F_X + \theta^\top \Phi^\top \Gamma \Phi \theta - 2F_X^\top \Gamma \Phi \theta \ . \tag{3.44}$$

On the other hand, since $\nabla_\theta J_X^w(\theta_X^w) = 0$, we have $\theta_X^{w\top}\Phi^\top\Gamma\Phi = F_X^\top\Gamma\Phi$. Thus, using the definition $\hat{\theta}_X^w = \theta_X^w + \eta$, a simple algebraic calculation yields

$$J_X^w(\hat{\theta}_X^w) - J_X^w(\theta_X^w) = \eta^\top\Phi^\top\Gamma\Phi\eta - F_X^\top\Gamma\Phi\eta - \eta^\top\Phi^\top\Gamma\Phi\theta_X^w \ . \tag{3.45}$$

Finally, taking the expectation over $\eta \sim \mathcal{N}(0, \sigma_X^2 I)$ of the above expression we get

$$\mathbb{E}_\eta\left[J_X^w(\hat{\theta}_X^w) - J_X^w(\theta_X^w)\right] = \mathbb{E}_\eta\left[\eta^\top\Phi^\top\Gamma\Phi\eta\right] = \sigma_X^2 \operatorname{Tr}(\Phi^\top\Gamma\Phi) = \sigma_X^2\|\Gamma^{1/2}\Phi\|_F^2 \ . \tag{3.46}$$

□

In order to bound $\mathbb{E}_X\left[\sigma_X^2\right]$ we recall the variance has the form $\sigma_X^2 = C^2\psi_X^w$, where C is a constant independent of X and

$$\psi_X^w = \max_{k\geq 0} e^{-k\beta} \sum_{s\in\mathcal{S}} \frac{w_s}{\max\{|X_s| - k, 1\}^2} \leq \sum_s w_s \left(\max_{k\geq 0} \frac{e^{-k\beta}}{\max\{|X_s| - k, 1\}^2}\right) \ . \tag{3.47}$$

Thus, we can bound $\mathbb{E}_X\left[\sigma_X^2\right] = C^2\mathbb{E}_X\left[\psi_X^w\right]$ by providing a bound for the expectation of each individual maximum in (3.47). The two following technical lemmas will prove useful.

Lemma 3.14. *Let $b > 0$ and $a \geq 1$. Then the following holds:*

$$\max_{0\leq x\leq a-1} \frac{e^{-bx}}{(a-x)^2} = \begin{cases} \frac{1}{a^2} & b < 2/a \\ e^{1-ab} & b > 2 \\ \frac{e^2}{4}b^2e^{-ab} & \text{otherwise} \end{cases} \tag{3.48}$$

Proof. The result follows from a simple calculation. □

Lemma 3.15. *Suppose $B_{m,p}$ is a binomial random variable with m trials and success probability p. Then the following hold:*

$$\mathbb{E}\left[\frac{1}{B_{m,p}+1}\right] = \frac{1-(1-p)^{m+1}}{p(m+1)} ,$$

$$\mathbb{E}\left[\frac{1}{B_{m,p}^2}\mathbb{I}_{B_{m,p}\geq 1}\right] \leq \frac{6}{p(m+1)}\left(\frac{1-(1-p)^{m+2}}{p(m+2)} - (1-p)^{m+1} - \frac{p(m+1)}{2}(1-p)^m\right) .$$

Proof. The first expectation is a classical exercise in probability textbooks. The second one can be proved as follows:

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{B_{m,p}^2}\mathbb{I}_{B_{m,p}\geq 1}\right] &= \sum_{k=1}^{m}\frac{1}{k^2}\binom{m}{k}p^k(1-p)^{m-k} \\
&\leq 6\sum_{k=1}^{m}\frac{1}{(k+1)(k+2)}\binom{m}{k}p^k(1-p)^{m-k} \\
&= \frac{6}{p(m+1)}\sum_{k=1}^{m}\frac{1}{k+2}\frac{(m+1)!}{(k+1)!(m-k)!}p^{k+1}(1-p)^{m-k} \\
&= \frac{6}{p(m+1)}\sum_{k=1}^{m}\frac{1}{k+2}\mathbb{P}[B_{m+1,p}=k+1] \\
&= \frac{6}{p(m+1)}\sum_{j=2}^{m+1}\frac{1}{j+1}\mathbb{P}[B_{m+1,p}=j] \\
&= \frac{6}{p(m+1)}\left(\mathbb{E}\left[\frac{1}{B_{m+1,p}+1}\right] - \mathbb{P}[B_{m+1,p}=0] - \frac{1}{2}\mathbb{P}[B_{m+1,p}=1]\right) \\
&= \frac{6}{p(m+1)}\left(\frac{1-(1-p)^{m+2}}{p(m+2)} - (1-p)^{m+1} - \frac{p(m+1)}{2}(1-p)^m\right) ,
\end{aligned}$$

where we used the first equation in the last step, and the bound $(k+1)(k+2)/k^2 \leq 6$ for $k \geq 1$ in the first inequality. □

Recall that p_s denotes the probability that a trajectory from X visits states s. Because these trajectories are i.i.d. we have that $|X_s| = B_{m,p_s}$ is a binomial random variable. Therefore, we can combine the last two lemmas to prove the following.

Lemma 3.16. *Suppose $\beta \leq 2$. Then we have:*

$$\mathbb{E}_X\left[\max_{k\geq 0}\frac{e^{-k\beta}}{\max\{|X_s|-k,1\}^2}\right] \leq \begin{cases}\frac{6}{p_s^2(m+1)(m+2)} + \frac{e^2\beta^2}{4}(1-(1-e^{-\beta})p_s)^m & p_s > 0\ ,\\ 1 & p_s = 0\ .\end{cases} \tag{3.49}$$

Proof. Note in the first place that Lemma 3.14 implies

$$\max_{k\geq 0}\frac{e^{-k\beta}}{\max\{|X_s|-k,1\}^2} = \mathbb{I}_{|X_s|=0} + \mathbb{I}_{1\leq|X_s|<2/\beta}\frac{1}{|X_s|^2} + \mathbb{I}_{|X_s|\geq 2/\beta}\frac{e^2}{4}\beta^2 e^{-\beta|X_s|}\ , \tag{3.50}$$

where we used that in the case $|X_s| = 0$ the maximum is 1. If $p_s = 0$, then obviously $|X_s| = 0$ almost surely and the expectation of (3.50) equals 1. On the other hand, when $p_s > 0$ we use the linearity of expectation and bound each term separately. Clearly, $\mathbb{E}_X\left[\mathbb{I}_{|X_s|=0}\right] = \mathbb{P}_X[B_{m,p_s} = 0] = (1-p_s)^m$. On the other hand, by looking up the moment generating function of a binomial distribution we have

$$\mathbb{E}_X\left[\mathbb{I}_{|X_s|\geq 2/\beta}\frac{e^2}{4}\beta^2 e^{-\beta|X_s|}\right] \leq \frac{e^2}{4}\beta^2\mathbb{E}_X\left[e^{-\beta|X_s|}\right] = \frac{e^2}{4}\beta^2(1-(1-e^{-\beta})p_s)^m\ . \tag{3.51}$$

The remaining term is bounded by

$$\mathbb{E}_X\left[\mathbb{I}_{1\leq|X_s|<2/\beta}\frac{1}{|X_s|^2}\right] \leq \mathbb{E}_X\left[\mathbb{I}_{1\leq|X_s|}\frac{1}{|X_s|^2}\right]\ . \tag{3.52}$$

Therefore, applying Lemma 3.15 and upper bounding some negative terms by zero, we get

$$\mathbb{E}_X\left[\max_{k\geq 0}\frac{e^{-k\beta}}{\max\{|X_s|-k,1\}^2}\right] \leq \frac{6}{p_s^2(m+1)(m+2)} + \frac{e^2\beta^2}{4}(1-(1-e^{-\beta})p_s)^m\ . \tag{3.53}$$

□

Now we can combine Lemmas 3.13 and 3.16 using Equation 3.47 to get our final result. This theorem bounds the expected empirical excess risk of DP-LSW. The bound contains two terms: one vanishes as $m \to \infty$, and the other reflects the fact that states which are never visited pose a problem to stability.

Theorem 3.17. *Let $\mathcal{S}_0 = \{s \in \mathcal{S} | p_s = 0\}$ and $S_+ = \mathcal{S} \setminus \mathcal{S}_0$.
Let $C = \alpha R_{\max} \|(\Gamma^{1/2}\Phi)^\dagger\| \|\Gamma^{1/2}\Phi\|_F / (1-\gamma)$. Suppose $\beta \leq 2$. Then we have the following:*

$$
\begin{aligned}
&\mathbb{E}_{X,\eta}\left[J_X^w(\hat{\theta}_X^w) - J_X^w(\theta_X^w)\right] \\
&\leq C^2 \left(\sum_{s \in \mathcal{S}_0} w_s + \sum_{s \in \mathcal{S}_+} w_s \left(\frac{6}{p_s^2 (m+1)(m+2)} + \frac{e^2 \beta^2}{4} (1 - (1 - e^{-\beta}) p_s)^m \right) \right) .
\end{aligned}
$$

By noting that $e^2/4 \leq 6$, $m^2 \leq (m+1)(m+2)$, and when $\beta \leq 1/2$ then $1 - (1 - e^{-\beta}) p_s \leq 1 - \beta p_s / 2$, then the following corollary is easily obtained.

Corollary 3.18. *Let $\mathcal{S}_0 = \{s \in \mathcal{S} | p_s = 0\}$ and $S_+ = \mathcal{S} \setminus \mathcal{S}_0$.
Let $C = \alpha R_{\max} \|(\Gamma^{1/2}\Phi)^\dagger\| \|\Gamma^{1/2}\Phi\|_F / (1-\gamma)$. Suppose $\beta \leq 1/2$.
Then $\mathbb{E}_{X,\eta}\left[J_X^w(\hat{\theta}_X^w) - J_X^w(\theta_X^w)\right]$ is upper bounded by:*

$$
C^2 \left(\sum_{s \in \mathcal{S}_0} w_s + 6 \sum_{s \in \mathcal{S}_+} w_s \left(\frac{1}{p_s^2 m^2} + \beta^2 \left(1 - \frac{\beta p_s}{2} \right)^m \right) \right) .
$$

Note the above bound depends on the dimension d through β and $\|\Gamma^{1/2}\Phi\|_F$. In terms of the size of the dataset, we can get excess risk bounds that decreases quadratically with m by assuming that either all states are visited with non-zero probability or the user sets the regression weights so that such states do not contribute to θ_X^w.

The following is an immediate consequence of the results in this section.

Corollary 3.19. *If $w_s = 0$ for all $s \in \mathcal{S}_0$, then $\mathbb{E}_{X,\eta}\left[J_X^w(\hat{\theta}_X^w) - J_X^w(\theta_X^w)\right] = O(1/m^2)$.*

A similar theorem can be proved for DP-LSL. However, in this case the statement of the bound is complicated by the appearance of co-occurrence probabilities of the form $\mathbb{P}_x[s \in x \wedge s' \in x]$ and $\mathbb{P}_x[s \in x \wedge s' \notin x]$.

3.7 Utility Analysis of DP-LSL

The analysis in this section follows a scheme similar to the previous one. We start by taking the expectation of the excess empirical risk with respect to the Gaussian perturbation η.

Lemma 3.20.

$$\mathbb{E}_{X,\eta}\left[J_X^\lambda(\hat{\theta}_X^\lambda) - J_X^\lambda(\theta_X^\lambda)\right] = \mathbb{E}_X\left[\left(\frac{\lambda d}{2m} + \frac{1}{m}\sum_{s\in\mathcal{S}} \rho_s \|\phi_s\|_2^2 |X_s|\right)\sigma_X^2\right] . \tag{3.54}$$

Proof. Let X be an arbitrary dataset with m trajectories. Recalling that $\hat{\theta}_X^\lambda = \theta_X^\lambda + \eta$ we get:

$$\begin{aligned}
&J_X^\lambda(\hat{\theta}_X^\lambda) - J_X^\lambda(\theta_X^\lambda)\\
&= \frac{1}{m}\sum_{i=1}^m \sum_{s\in\mathcal{S}_{x_i}} \rho_s \left((\phi_s^\top \hat{\theta}_X^\lambda)^2 - (\phi_s^\top \theta_X^\lambda)^2 - 2F_{x_i,s}\phi_s^\top \eta\right) + \frac{\lambda}{2m}\left(\|\hat{\theta}_X^\lambda\|_2^2 - \|\theta_X^\lambda\|_2^2\right)\\
&= \frac{1}{m}\sum_{i=1}^m \sum_{s\in\mathcal{S}_{x_i}} \rho_s \left(\eta^\top \phi_s\phi_s^\top \eta + 2\eta^\top \phi_s\phi_s^\top \theta_X^\lambda - 2F_{x_i,s}\phi_s^\top \eta\right) + \frac{\lambda}{2m}\left(\|\eta\|_2^2 + 2\eta^\top \theta_X^\lambda\right) .
\end{aligned}$$

Taking the expectation over $\eta \sim \mathcal{N}(0, \sigma_X^2 I)$ in the above expression we get

$$\mathbb{E}_\eta\left[J_X^\lambda(\hat{\theta}_X^\lambda) - J_X^\lambda(\theta_X^\lambda)\right] = \frac{1}{m}\sum_{i=1}^m \sum_{s\in\mathcal{S}_{x_i}} \rho_s \operatorname{Tr}(\phi_s\phi_s^\top)\sigma_X^2 + \frac{\lambda}{2m} d\sigma_X^2 .$$

The result now follows from noting that $\sum_{i=1}^m \sum_{s\in\mathcal{S}_{x_i}} \rho_s \operatorname{Tr}(\phi_s\phi_s^\top) = \sum_{s\in\mathcal{S}} \rho_s \|\phi_s\|_2^2 |X_s|$. □

In order to bound the expression given by previous lemma we will expand the definition of $\sigma_X = C_\lambda\sqrt{\psi_X^\lambda}$, with $C_\lambda = 2R_{\max}\|\Phi\|/(1-\gamma)(\lambda - \|\Phi\|^2\|\rho\|_\infty)$, and note that using the straightforward bound $(a+b)^2 \le 2a^2 + 2b^2$ we have:

$$\begin{aligned}
\psi_X^\lambda &= \max_{k\ge 0} e^{-k\beta}\left(\|\rho\|_2 + \frac{\|\Phi\|\|\rho\|_\infty}{\sqrt{2\lambda}}\sqrt{\sum_{s\in\mathcal{S}} \rho_s \min\{|X_s| + k, m\}}\right)^2\\
&\le 2\|\rho\|_2^2 + \frac{\|\Phi\|^2\|\rho\|_\infty^2}{\lambda}\sum_{s\in\mathcal{S}} \rho_s \max_{k\ge 0}\left(e^{-2k\beta}\min\{|X_s| + k, m\}\right) .
\end{aligned}$$

The following lemma can be used to bound the maximums inside this sum.

Lemma 3.21. *Suppose $a \ge 0$ and $b > 0$. Then the following holds:*

$$\max_{0\le x\le m-a} e^{-2bx}(a+x) = \begin{cases} a & b < a/2 \\ me^{-2b(m-a)} & b > m/2 \\ \frac{1}{2eb}e^{2ab} & \textit{otherwise} \end{cases} \tag{3.55}$$

Assuming we have $2\beta < 1 \leq m$, previous lemma yields:

$$\max_{k \geq 0} \left(e^{-2k\beta} \min\{|X_s| + k, m\} \right) = |X_s| \mathbb{I}_{|X_s| > 2\beta} + \frac{1}{2e\beta} e^{2\beta |X_s|} \mathbb{I}_{|X_s| \leq 2\beta} \leq |X_s| + \frac{1}{2e\beta} \mathbb{I}_{|X_s| = 0} . \tag{3.56}$$

When taking the expectation of the upper bound for (3.54) obtained by plugging in (3.56), several quantities involving products of correlated binomial random variables will appear. Next lemma gives expressions for all these expectations.

Lemma 3.22. *Recall that $p_s = \mathbb{P}[s \in x]$ and $|X_s|$ is a binomial random variable with m trials and success probability p_s. Define $p_{s,s'} = \mathbb{P}[s \in x \wedge s' \in x]$ and $\bar{p}_{s,s'} = \mathbb{P}[s \in x \wedge s' \notin x]$ for any $s, s' \in \mathcal{S}$. Then we have the following:*

1. $\mathbb{E}[|X_s|] = m p_s$,
2. $\mathbb{E}\left[\mathbb{I}_{|X_s|=0}\right] = (1 - p_s)^m$,
3. $\mathbb{E}\left[|X_s|^2\right] = m^2 p_s^2 + m(p_s - p_s^2)$,
4. $\mathbb{E}[|X_s||X_{s'}|] = m(m-1) p_s p_{s'} + m p_{s,s'}$,
5. $\mathbb{E}\left[|X_s| \mathbb{I}_{|X_{s'}|=0}\right] = m \bar{p}_{s,s'} (1 - p_{s'})^{m-1}$.

Proof. All equations follow from straightforward calculations. □

Theorem 3.23. *Suppose $\beta < 1/2$ and $\lambda > \|\Phi\|^2 \|\rho\|_\infty$. Let $C_\lambda = 2\alpha R_{\max} \|\Phi\| / (1-\gamma)(\lambda - \|\Phi\|^2 \|\rho\|_\infty)$. Then we have*

$$\begin{aligned}
&\mathbb{E}_{X,\eta}\left[J_X^\lambda(\hat{\theta}_X^\lambda) - J_X^\lambda(\theta_X^\lambda) \right] \\
&\leq C_\lambda^2 \Bigg\{ \sum_{s \in \mathcal{S}} \rho_s p_s \left(\frac{d \|\Phi\|^2 \|\rho\|_\infty^2}{2} + 2 \|\rho\|_2^2 \|\phi_s\|_2^2 \right) \\
&+ \frac{\lambda}{m} d \|\rho\|_2^2 + \frac{1}{m} \frac{d \|\Phi\|^2 \|\rho\|_\infty^2}{4e\beta} \sum_{s \in \mathcal{S}} \rho_s (1 - p_s)^m + \frac{m}{\lambda} \|\Phi\|^2 \|\rho\|_\infty^2 \sum_{s,s' \in \mathcal{S}} \rho_s \rho_{s'} p_s p_{s'} \|\phi_s\|_2^2 \\
&+ \frac{1}{\lambda} \|\Phi\|^2 \|\rho\|_\infty^2 \Bigg(\sum_{s \in \mathcal{S}} \rho_s^2 \|\phi_s\|_2^2 (p_s - p_s^2) \\
&+ \sum_{\substack{s,s' \in \mathcal{S} \\ s \neq s'}} \rho_s \rho_{s'} \|\phi_s\|_2^2 \left(p_{s,s'} - p_s p_{s'} + \frac{1}{2e\beta} \bar{p}_{s,s'} (1 - p_{s'})^{m-1} \right) \Bigg) \Bigg\} .
\end{aligned}$$

Proof. Combining Lemma 3.20 with (3.56) and the definition of σ_X^2 yields the following upper bound for $\mathbb{E}_{X,\eta}\left[J_X^\lambda(\hat{\theta}_X^\lambda) - J_X^\lambda(\theta_X^\lambda) \right]$:

$$C_\lambda^2 \mathbb{E}_X \left[\left(\frac{\lambda d}{2m} + \frac{1}{m} \sum_{s \in \mathcal{S}} \rho_s \|\phi_s\|_2^2 |X_s| \right) \left(2\|\rho\|_2^2 + \frac{\|\Phi\|^2 \|\rho\|_\infty^2}{\lambda} \sum_{s \in \mathcal{S}} \rho_s \left(|X_s| + \frac{1}{2e\beta} \mathbb{I}_{|X_s|=0} \right) \right) \right] .$$

Terms that do not involve products of the form $|X_s||X_{s'}|$ or $|X_s|\mathbb{I}_{|X_{s'}|=0}$ can be straightforwardly reduced to linear combinations of expectations in Lemma 3.22. The remaining term yields the following:

$$\begin{aligned}
&\mathbb{E}_X\left[\sum_{s,s'\in\mathcal{S}}\rho_s\rho_{s'}\|\phi_s\|_2^2|X_s|\left(|X_{s'}|+\frac{1}{2e\beta}\mathbb{I}_{|X_{s'}|=0}\right)\right]\\
&=\sum_{s\in\mathcal{S}}\rho_s^2\|\phi_s\|_2^2\mathbb{E}_X\left[|X_s|\left(|X_s|+\frac{1}{2e\beta}\mathbb{I}_{|X_s|=0}\right)\right]\\
&\quad+\sum_{\substack{s,s'\in\mathcal{S}\\ s\neq s'}}\rho_s\rho_{s'}\|\phi_s\|_2^2\mathbb{E}\left[\right]X\left[|X_s|\left(|X_{s'}|+\frac{1}{2e\beta}\mathbb{I}_{|X_{s'}|=0}\right)\right]\\
&=\sum_{s\in\mathcal{S}}\rho_s^2\|\phi_s\|_2^2\left(m^2p_s^2+m(p_s-p_s^2)\right)\\
&\quad+\sum_{\substack{s,s'\in\mathcal{S}\\ s\neq s'}}\rho_s\rho_{s'}\|\phi_s\|_2^2\left(m(m-1)p_sp_{s'}+mp_{s,s'}+\frac{1}{2e\beta}m\bar{p}_{s,s'}(1-p_{s'})^{m-1}\right)\ ,
\end{aligned}$$

where we used Lemma 3.22 again. Thus we get:

$$\begin{aligned}
&\mathbb{E}_{X,\eta}\left[J_X^\lambda(\hat{\theta}_X^\lambda)-J_X^\lambda(\theta_X^\lambda)\right]\le C_\lambda^2\left\{\frac{\lambda d\|\rho\|_2^2}{m}+\frac{d\|\Phi\|^2\|\rho\|_\infty^2}{2m}\sum_{s\in\mathcal{S}}\rho_s\left(mp_s+\frac{1}{2e\beta}(1-p_s)^m\right)\right.\\
&+\frac{2\|\rho\|_2^2}{m}\sum_{s\in\mathcal{S}}\rho_sp_s\|\phi_s\|_2^2m+\frac{\|\Phi\|^2\|\rho\|_\infty^2}{\lambda m}\sum_{s\in\mathcal{S}}\rho_s^2\|\phi_s\|_2^2\left(m^2p_s^2+m(p_s-p_s^2)\right)\\
&\left.+\frac{\|\Phi\|^2\|\rho\|_\infty^2}{\lambda m}\sum_{\substack{s,s'\in\mathcal{S}\\ s\neq s'}}\rho_s\rho_{s'}\|\phi_s\|_2^2\left(m(m-1)p_sp_{s'}+mp_{s,s'}+\frac{1}{2e\beta}m\bar{p}_{s,s'}(1-p_{s'})^{m-1}\right)\right\}
\end{aligned}$$

The final result is obtained by grouping the terms in this expression by their dependence in λ and m. □

Note that if we take $\lambda=\omega(1)$ with respect to m in the above theorem, then $C_\lambda=O(1/\lambda)$ and we get the following corollary.

Corollary 3.24. *Suppose $\lambda=\omega(1)$ with respect to m. Then we have*

$$\mathbb{E}_{X,\eta}\left[J_X^\lambda(\hat{\theta}_X^\lambda)-J_X^\lambda(\theta_X^\lambda)\right]=O\left(\frac{1}{\lambda m}+\frac{1}{\lambda^2}+\frac{m}{\lambda^3}\right)\ . \tag{3.57}$$

The Corollary 3.24 is obtained by assuming the regularization parameter is allowed to grow with m, and stresses the tensions in selecting an adequate regularization schedule.

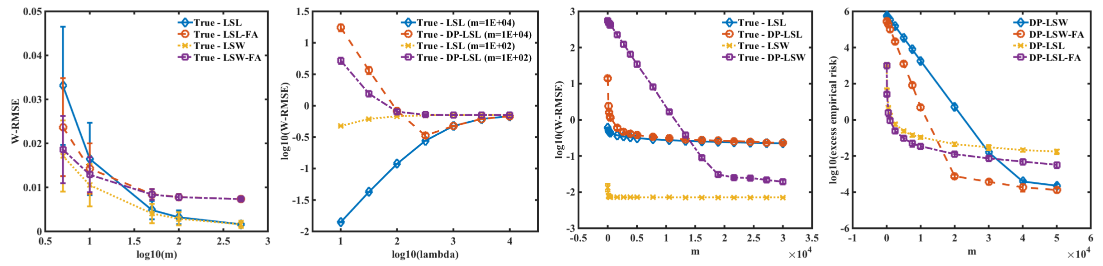


FIGURE 3.1: Empirical comparison of differentially private and non-private algorithms

Note that taking $\lambda = \Theta(m)$ we get a bound on the excess risk of order $O(1/m^2)$. However, if we want the regularization term in $J_X^\lambda(\theta)$ to vanish as $m \to \infty$ we need $\lambda = o(m)$. We shall see importance of this trade-off in our experiments.

3.8 Experiments

In this section we illustrate the behaviour of the proposed algorithms on synthetic examples. The domain we use consists of a chain of N states, where in each state the agent has some probability p of staying and probability $(1-p)$ of advancing to its right. There is a reward of 1 when the agent reaches the final, absorbing state, and 0 for all other states. While this is a toy example, it illustrates the typical case of policy evaluation in the medical domain, where patients tend to progress through stages of recovery at different speeds, and past states are not typically revisited (partly because in the medical domain, states contain historic information about past treatments). Trajectories are drawn by starting in an initial state distribution and generating state-action-reward transitions according to the described probabilities until the absorbing state is reached. Trajectories are harvested in a batch, and the same batches are processed by all algorithms.

We experiment with both a tabular representation of the value function, as well as with function approximation. In the latter case, we simply aggregate pairs of adjacent states, which are hence forced to take the same value. We compared the proposed private algorithms DP-LSW and DP-LSL with their non-private equivalents LSW and LSL. The performance measure used is average root mean squared error over the state space. The error is obtained by comparing the state values estimated by the learning algorithms against the exact values obtained by exact, tabular dynamic programming. Standard errors computed over 20 independent runs are included.

The main results are summarized in Fig. 3.1, for an environment with $N = 40$ states, $p = 0.5$, discount $\gamma = 0.99$, and for the DP algorithms, $\varepsilon = 0.1$ and $\delta = 0.1$. In general, these constants should be chosen depending on the privacy constraints of the domain. Our theoretical results explain the expected effect of these choices on the privacy-utility trade-off so we do not provide extensive experiments with different values.

The w_s parameters in LSW algorithm can be used to influence the relative importance of the value estimate in different states. They play a similar role to the ρ_s in DP-LSL algorithm, but we gave them a different name because as described above there is a choice of w_s that mimics the asymptotic behaviour of LSL algorithm with given ρ_s. If no prior knowledge is available, the w_s can be chosen to be uniform (as we do in our experiments). Corollary 3.24 in our utility analysis shows that DP-LSL algorithm behaves nicely if these weights satisfy a mild condition.

The left plot in Fig. 3.1 compares the non-private LSL and LSW versions of Monte Carlo evaluation, in the tabular and function approximation case. As can be seen, both algorithms are very stable and converge to the same solution, but LSW converges faster. The second plot compares the performance of all algorithms in the tabular case, over a range of regularization parameters, for two different batch sizes. The third plot compares the expected RMSE of the algorithms when run with state aggregation, as a function of batch size. As can be seen, the DP algorithms converge to the same solutions as the non-private corresponding versions for large enough batch sizes. Interestingly, the two proposed approaches serve different needs. The LSL algorithms work better with small batches of data, whereas the LSW approach is preferable with large batches. From an empirical point of view, the trade-off between accuracy and privacy in the DP-LSL algorithm should be done by setting a regularization schedule proportional to $\sqrt{m}$. While the theory suggests it is not the best schedule in terms of excess empirical risk, it achieves the best overall accuracy.

Finally, the last figure shows excess empirical risk as a function of the batch size. Interestingly, more aggressive function approximation helps both differentially private algorithms converge faster. This is intuitive, since using the same data to estimate fewer parameters means the effect of each individual trajectory is already obscured by the function approximation. Decreasing the number of parameters of the function approximator, d, increases β, which lowers the smooth sensitivity bounds. In medical applications, one expects to have many attributes measured about patients, and to need aggressive function approximation in order to provide generalization. This result tells us that differentially private algorithms should be favoured in this case as well.

Overall, the empirical results are very promising, showing that especially as batch size increases, the noise introduced by the DP mechanism decreases rapidly, and these algorithms provide the same performance but with the additional privacy guarantees.

3.9 Discussion

In this chapter, we presented the first (to our knowledge) differentially private algorithms for policy evaluation in the full MDP setting. Our algorithms are built on top of established Monte Carlo methods, and come with utility guarantees showing that the cost of privacy diminishes as training batches get larger. The smoothed sensitivity framework is a key component of our analyses, which differ from previous works on DP mechanisms for ERM and bandits problems in two substantial ways. First, we consider optimizations with non-Lipschitz loss functions, which prevents us from using most of the established techniques for analyzing privacy and utility in ERM algorithms and complicates some parts of our analysis. In particular, we cannot leverage the tight utility analysis of (Jain and Thakurta, 2014) to get dimension independent bounds. Second, and more importantly, the natural model of neighbouring datasets for policy evaluation involves replacing a whole trajectory. This implies that neighbouring datasets can differ in multiple regression targets, which is quite different from the usual supervised learning approach where neighbouring datasets can only change a single regression target. Our approach is also different from the on-line learning and bandits setting, where there is a single stream of experience and neighbouring datasets differ in one element of the stream. Note that this setting cannot be used naturally in the full MDP setup, because successive observations in a single stream are inherently correlated.

Our techniques could be extended in two directions. First, it would be interesting to design differentially private policy evaluation methods based on temporal-difference learning methods (Sutton, 1988). This case is difficult because regression targets become non-stationary, depending on the current value function, so the methodology we used in the Monte Carlo case does not readily apply. One path to obtain results in this case is to utilize the least-squares temporal-difference learning algorithm (Boyan, 2002; Ghavamzadeh et al., 2010), which is a model-based algorithm. In its linear version, it estimates the expected feature-to-feature transition model and the reward vector, then computes the value function by inverting the transition matrix and multiplying by the reward vector. This algorithm has the same fixed point as usual temporal-difference learning, but the learning of the model is much more similar to the supervised case. We actually pursued this line of research for some time, but in the end, the structure of the problem is still too different from supervised learning overall and we were not able to obtain a satisfactory result of the type presented above for the Monte Carlo case. We suspect that a different approach to perturbation will be necessary to tackle this type of algorithms.

Secondly, it would be important to tackle the control case, where policy evaluation is often used as a sub-routine, e.g. as in actor-critic methods. This case is even more complex than temporal-difference learning, because the actions taken by the agent influence the data that will be seen in the future. This complication is not present in bandit algorithms, where most differential privacy for control has been studied. There, an action only influences the reward, not the next observed state, so privacy analysis is a bit easier to carry out. It is not clear at the moment how to take both the differential privacy bandit results from the literature and the results we presented in this chapter and bridge them in order to be able to tackle the MDP case.

Finally, it would also be important to to evaluate the algorithms we proposed in this chapter on real data. For example, patient data from clinical studies would potentially create a compelling case study, with the caveat that errors could not be estimated precisely, because the right answer is not known. As we were not able to secure such data so far, we leave this direction open for future work.

Chapter 4

Utility Amplification via Sub-Sampling

In the design and analysis of differentially private mechanisms, sub-sampling techniques have become fundamental cornerstones. While the existing literature mostly focuses on privacy amplification (Balle et al., 2018; Wang et al., 2019), the literature on the utility of sub-sampling-based methods is relatively unexplored. Specifically, in the setting, that privacy-preserving mechanism is only available for a fixed privacy regime. In such setting , knowing how to add an intermediate step or subroutine to the algorithm to boost the accuracy of the original private algorithm is of great value. This chapter's main motivation is to initiate a systematic study of utility boosting of the private algorithm by sub-sampling. In this chapter, we provide the necessary tools for a systematic analysis of the impact of sub-sampling on the utility of private linear queries. With the foundation established in Chapter 3, we now have the necessary background to the challenges in designing private policy evaluation algorithms in reinforcement learning. Therefore, this chapter's main focus is to provide a utility boosting subroutine for the existing private policy evaluation algorithms.

In this chapter, we begin in Section 4.1 by providing a broad overview of different frameworks that employ sub-sampling subroutine in the context of privacy-preserving algorithm design. In Section 4.2 we provide the necessary formal foundation to develop a new sub-sampling subroutine to improve the results of algorithms proposed in Chapter 3. We propose new notions of sensitivity and utility tailored to the proposed framework. In Section 4.3 we present the utility analysis of classic multivariate Gaussian output perturbation mechanism based on the new notions of sensitivity and utility introduced in Section 4.2. Subsequently, in Section 4.6 we analyze the average utility of the proposed sub-sample and average mechanism for linear queries. Finally, since the DP-LSL and DP-LSW algorithms proposed in Chapter 3 are linear function approximators and thus resemble a form of linear query, we apply the framework proposed in Section 4.6 on DP-LSL and DP-LSW algorithms and propose a new algorithm that boosts their utility. We then experimentally examine the performance of proposed algorithm based on the private policy evaluation algorithms proposed in 3 and we show a significant improvement over the baseline algorithms.

4.1 Review of Sub-Sampling Techniques Used in Differential Privacy

The impact of sub-sampling on privacy amplification has recently gained more attention compared to other privacy amplification techniques such as iteration (Feldman et al., 2018), composition (Kairouz et al., 2015; Vadhan, 2017) and shuffling (Erlingsson et al., 2019; Balle et al., 2019).

One of the early attempts in applying sub-sampling subroutines in the context of privacy-preserving algorithm is *Sub-sample and aggregate*(SA) framework proposed by Nissim et al. (2007). SA framework is designed to resolve the *worst-case sensitivity* analysis problem as one of the main obstacles in designing DP algorithms. Particularly, output perturbation mechanisms such as Laplacian or Gaussian add noise proportional to the target algorithm sensitively. However, sub-sample and aggregate framework benefits from the assumption that if the target algorithm yields accurate output on the input data x with high probability, it should yield proportionally accurate output on a random sub-sample of x. The SA is proposed in the setting where the sub-sampling is done without replacement (WOR) and the data is partitioned into equisized disjoint mini batches.

Bootstrap aggregating or bagging (Breiman, 1996) can be seen as an extension of SA technique and is known as a classic machine learning technique that improves stability and accuracy of underlying statistical classifier or regressor (Freund et al., 1996; Quinlan et al., 1996). Recently, Jordon et al. (2019) proposed DP bagging method that benefits from a methodological difference between bagging and SA framework. In Jordon et al. (2019) the bagging framework proposed first duplicates the input dataset, then partitions each instance into equisized disjoint mini batches and then pass each mini batch to private estimators.

The sub-sampling technique we use in this chapter is more along the same line as studied in Balle et al. (2018); Wang et al. (2019), where instead of partitioning the input dataset into a set of disjoint sub-samples (eg. (Nissim et al., 2007)) or replicating multiple instances of the input detaset and then use the classic bagging technique (eg. (Jordon et al., 2019)), we arbitrarily or adaptively choose the sub-sample size and the number of sub-samples and generate finite number of sub-samples and then run the private algorithm on each sub-sample. In this method sub-sampling is often done uniformly with or without replacement and thus the mini batches are not disjoint. In Figure 4.1 we present the schematical difference between the sub-sampling

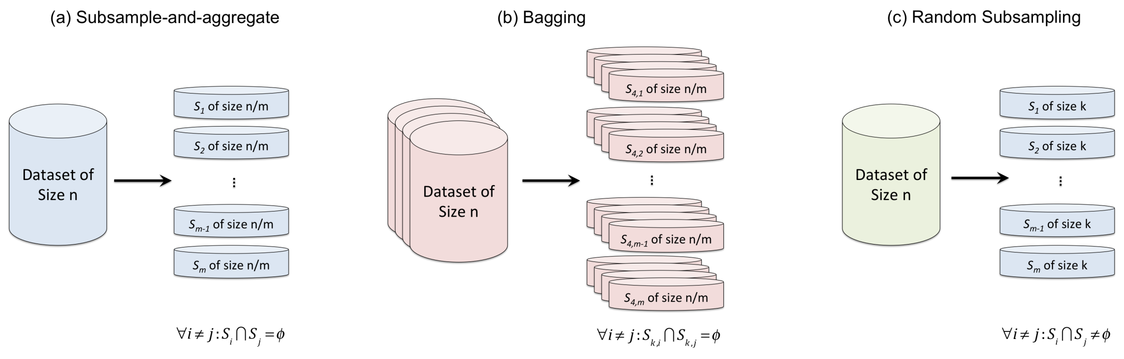


FIGURE 4.1: Schematic of different sub-sampling methods: (a) subsample-and-aggregate framework (Nissim et al., 2007); (b) Differntial bagging framework (Jordon et al., 2019); (c) random sub-sampling method used in this chapter.

method that we adopt in this chapter and the other commonly used sub-sampling techniques used in the literature.

Since the sub-sample size and number of sub-samples in our method are chosen arbitrarily, two important questions are raised:

1. Given the input dataset size and the privacy parameters, what is the optimal number of sub-samples?
2. Given the input dataset size and the privacy parameters, what is the best choice of sub-sample size?

To the best of our knowledge, this is the first study that proposes a principled approach in analyzing optimal sub-sampling parameters in the context of differential private sub-sampling from the utility standpoint.

The contributions of this chapter are:

- Rigorous analysis of impact of sub-sampling on utility of differential private linear queries.
- Proposing adaptive DP algorithm that adaptively chooses the main sub-sampling parameters sub-sample size k and number of sub-samples m.

4.2 Notation and Framework

This section presents background and notation required for designing and analyzing sub-sampling-based framework we propose in this chapter.

We start by providing necessary definitions. A case of particular interest are *linear queries*, where the query is uniquely identified by a parameter vector ω and is defined as the following.

Definition 4.1 (Linear Query)**.** A linear query A_ω is uniquely identified by a parameter vector ω as $A_\omega(.) = \langle \omega, . \rangle$.

It is straightforward to show that, in the case that A is a linear query the global sensitivity of A satisfies $\mathrm{GS}_p^{(n)}(A) = \mathrm{GS}_p^{(1)}(A)/n$.

In the classic output perturbation-based mechanisms the upper-bound on the effect of data inclusion and exclusion is captured by the notion of global sensitivity, defined in definition 2.2. However, to capture the maximum impact of sub-sampling on the output of a randomized algorithm A in the following definition we introduce the notion of *global sub-sampling sensitivity*.

Definition 4.2 (Global Sub-sampling Sensitivity)**.** Given $x \in \mathbb{X}^n$ we denote by $\tilde{x} \sim_k x$ a random subset of x with $k \leq n$ elements uniformly sampled without replacement. We define the *global sub-sampling sensitivity* of A as,

$$\mathrm{GSS}_p^{(n,k)}(A) := \left(\sup_{x \in \mathbb{X}^n} \mathbb{E}_{\tilde{x} \sim_k x} \left[\| A(x) - A(\tilde{x}) \|_p^p \right] \right)^{1/p} . \tag{4.1}$$

Similarly to capture the average impact of ata inclusion and exclusion on the output of a randomized algorithm A we define *Average Sub-sampling Sensitivity*. Given $\mathcal{D}$ be a probability distribution over $\mathbb{X}$ in the following we define the *average sub-sampling sensitivity* of A.

Definition 4.3 (Average Sub-sampling Sensitivity)**.** Let $\mathcal{D}$ be a probability distribution over $\mathbb{X}$. The *average sub-sampling sensitivity* of A is defined as

$$\mathrm{AVSS}_p^{(n,k,\mathcal{D})}(A) := \left(\mathbb{E}_{x \sim \mathcal{D}^n} \left[\mathbb{E}_{\tilde{x} \sim_k x} \left[\| A(x) - A(\tilde{x}) \|_p^p \right] \right] \right)^{1/p} . \tag{4.2}$$

To fully capture the impact of sub-sampling on the utility of randomized algorithm A in the following definition we define the notion of *Average Sub-sampling Bias*, which captures the bias induced by the sub-sampling process.

Definition 4.4 (Average Sub-sampling Bias)**.** Let $\mathcal{D}$ be a probability distribution over $\mathbb{X}$ and k be the size of mini-batch such that each data point is sampled from $\mathcal{D}$. The *Average Sub-sampling Bias* of A is defined as,

$$\mathrm{ASB}_p^{(n,k,\mathcal{D})}(A) := \left(\mathbb{E}_{x \sim \mathcal{D}^n} \left[\| A(x) - \mathbb{E}_{\tilde{x} \sim_k x} \left[A(\tilde{x}) \right] \|_p^p \right] \right)^{1/p} . \tag{4.3}$$

Utility measure: We now move on to define the utility measure that captures the impact of randomization sources. We aim to define a utility measure in order to,

1. compare two DP mechanisms for a fixed distribution,
2. compare the performance of one mechanism against another mechanism for all distributions.

Let $M = \{M_n\}_{n\in\mathbb{N}}$ be a family of (ε, δ)-DP mechanisms with inputs in $\mathbb{X}^n$ and output in $\mathbb{R}^d$. We say that M is an *output perturbation* mechanism for A if for any x we have $\mathbb{E}_M\left[M(x)\right] = A(x)$, where the expectation is taken over the randomization in the mechanism M. In other words, we consider how well the parameter vectors can be estimated under the output perturbation model. Thus in our proposed notion of utility we only focus on the variation of sub-sample and aggregate mechanism. Note that base of this notion of utility has been extensively used in the literature (Wang, 2018; Wasserman, 2013).

Definition 4.5 (Average Utility)**.** The *average utility* of M is defined as

$$\mathrm{UT}_p^{(n,\mathcal{D})}(M) := \mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_M\left[\|A(x) - M(x)\|_p^p\right]\right] \ , \tag{4.4}$$

where the internal expectation is over the randomization in M.

Now we are ready to define the *Sub-sample-and-Average Mechanism*, in which the choice of aggregation is empirical averaging.

Definition 4.6 (Sub-sample-and-Average Mechanism)**.** Suppose that $\tilde{M}$ is an output perturbation $(\tilde{\varepsilon}, \tilde{\delta})$-DP mechanism with inputs in $\mathbb{X}^n$ and outputs in $\mathbb{R}^d$. The *subsample-and-average* mechanism for $\tilde{M}^{(m,k)}$ is defined as

$$\tilde{M}^{(m,k)}(x) := \frac{1}{m}\sum_{i=1}^{m}\tilde{M}(\tilde{x}_i) \ , \qquad \tilde{x}_i \sim_k x \ . \tag{4.5}$$

4.3 Average Utility Case Study: Gaussian Output Perturbation Mechanism

We start our utility analysis by measuring the average utility of Gaussian output perturbation mechanism Dwork et al. (2014) for linear queries. For completeness, we first provide the Gaussian mechanism in the following theorem.

Theorem 4.7 (Gaussian output perturbation mechanism (Dwork et al., 2014))**.** *The output perturbation mechanism $M(.) = A(.) + \eta$ with noise parameters $\varepsilon, \delta \in (0, 1)$ and $\eta \sim \mathcal{N}(0, \sigma^2 I)$, where $\sigma^2 = 2\mathrm{GS}_2^{(n)}(A)^2\log(1.25/\delta)/\varepsilon^2$ is (ε, δ)-DP.*

We start by calculating the average sub-sampling bias of linear queries.

Lemma 4.8. *Let A be a linear query then* $\mathrm{ASB}_p^{(n,k,\mathcal{D})}(A) = 0$.

Proof. For linear queries we always have $\mathbb{E}_{\tilde{x}\sim_k x}\left[A(\tilde{x})\right] = A(x)$, thus we have,

$$\begin{aligned}\mathrm{ASB}_p^{(n,k,\mathcal{D})}(A) &= \left(\mathbb{E}_{x\sim\mathcal{D}^n}\left[\|A(x) - \mathbb{E}_{\tilde{x}\sim_k x}\left[A(\tilde{x})\right]\|_p^p\right]\right)^{1/p}\\ &= \left(\mathbb{E}_{x\sim\mathcal{D}^n}\left[\|A(x) - A(x)\|_p^p\right]\right)^{1/p} = 0\end{aligned}$$

□

In the following theorem we show the average utility of Gaussian output perturbation mechanism.

Theorem 4.9. *Let $A(.)$ be a linear query and $M(x) = A(x) + \eta$ with $\eta \sim \mathcal{N}(0, \sigma_n^2 I)$ be a (ε, δ)-DP Gaussian mechanism, then the average utility of M is,*

$$\mathrm{UT}_2^{(n,\mathcal{D})}(M) = C\frac{d\mathrm{GS}_2^{(1)}(A)^2\log(1/\delta)}{n^2\varepsilon^2}\ , \tag{4.6}$$

where $\sigma_n^2 = C\mathrm{GS}_2^{(n)}(A)^2\log(1/\delta)/\varepsilon^2$ and $C > 2\ln(1.25/\delta)$.

Proof. Suppose that $M(x) = A(x) + \eta$ with $\eta \sim \mathcal{N}(0, \sigma_n^2 I)$ is a (ε, δ)-DP Gaussian mechanism, so that $\sigma_n^2 = C\mathrm{GS}_2^{(n)}(A)^2\log(1/\delta)/\varepsilon^2$ (Dwork et al., 2014). Then we can calculate the average utility of M as

$$\begin{aligned}\mathrm{UT}_2^{(n,\mathcal{D})}(M) &= \mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_M\left[\|A(x) - M(x)\|_2^2\right]\right]\\ &= \mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_M\left[\|\eta\|_2^2\right]\right]\\ &= \mathbb{E}_M\left[\|\eta\|_2^2\right]\\ &= d\sigma_n^2\\ &= C\frac{d\mathrm{GS}_2^{(n)}(A)^2\log(1/\delta)}{\varepsilon^2}\ .\end{aligned}$$

Recall that if A is a linear query, then $\mathrm{GS}_2^{(n)}(A) = \mathrm{GS}_2^{(1)}(A)/n$, in which case we get

$$\mathrm{UT}_2^{(n,\mathcal{D})}(M) = C\frac{d\mathrm{GS}_2^{(1)}(A)^2\log(1/\delta)}{n^2\varepsilon^2}\ ,$$

with $C > 2\log 1.25/\delta$ we finally have,

$$\mathrm{UT}_2^{(n,\mathcal{D})}(M) = \frac{2\log(1.25/\delta)d\mathrm{GS}_2^{(1)}(A)^2\log(1/\delta)}{n^2\varepsilon^2}\ .$$

□

4.4 Privacy of Sub-Sample-and-Average Gaussian Mechanism

Our final goal is to compare the utility of the Gaussian mechanism M with that of the subsampled-and-averaged Gaussian mechanism $\tilde{M}^{(m,k)}$. Since for the comparison to be meaningful, we must achieve the same level of privacy on both mechanisms; we need to tune the privacy parameters of the base mechanism $\tilde{M}$ so that $\tilde{M}^{(m,k)}$ attains the desired level of privacy. To do so, for the sake of completeness, we re-state the following sub-sampling lemma.

Lemma 4.10 (Sub-sampling Lemma (Ullman, 2017))**.** *Let $x \in \mathbb{X}^n$, and $k \in \{0, 1, ..., n\}$, we denote by $\tilde{x} \sim_k x$ a random sub-sample of x with $k \leq n$ elements sampled without replacement. If $A(x)$ is (ε, δ)-DP, then mechanism $M_k(A(\tilde{x} \sim_k x))$, which runs A on $\tilde{x} \sim_k x$ is $(\frac{k(e^{\varepsilon}-1)}{n}, \frac{k\delta}{n})$-DP.*

Using the result of Lemma 4.10 and advance composition theorem from Dwork et al. (2010) we obtain the following theorem.

Theorem 4.11. *Suppose $A : \mathbb{X}^n \to \mathbb{R}^d$ is a linear query. Let $\tilde{M}$ be the (ε, δ)-DP Gaussian mechanism for A and $\tilde{M}^{(m,k)}$ the subsample-and-average mechanism for A. For any $\delta' > 0$ the mechanism $\tilde{M}^{(m,k)}$ is $(\varepsilon^*, \delta^*)$-DP with*

$$\varepsilon^* = \left(\frac{mk}{n} e^{\varepsilon}(e^{\varepsilon} - 1) + \sqrt{2m \log(1/\delta')}\right) \log\left(1 + \frac{k}{n} e^{\varepsilon}(e^{\varepsilon} - 1)\right) \ , \tag{4.7}$$

$$\delta^* = \delta \frac{mk}{n} e^{\varepsilon} + \delta' \ , \tag{4.8}$$

or,

$$\varepsilon = \log\left(\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{n\varepsilon^*}{k\left(\sqrt{8m \log(1/\delta')}\right)}}\right) \ , \tag{4.9}$$

$$\delta = \frac{n(\delta^* - \delta')}{mk} \frac{1}{\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{n}{k} \frac{\varepsilon^*}{\sqrt{8m \log(\frac{1}{\delta'})}}}} \ . \tag{4.10}$$

Proof. A simple analysis combining the results from lemma 4.10 together with the advance composition theorem from Dwork et al. (2014) shows that $\tilde{M}^{(m,k)}$ is $(\varepsilon^*, \delta^*)$-DP with

$$\varepsilon^* = \left(\frac{mk}{n} e^{\tilde{\varepsilon}}(e^{\tilde{\varepsilon}} - 1) + \sqrt{2m \log(1/\delta')}\right) \log\left(1 + \frac{k}{n} e^{\tilde{\varepsilon}}(e^{\tilde{\varepsilon}} - 1)\right) \ ,$$

$$\delta^* = \tilde{\delta} \frac{mk}{n} e^{\tilde{\varepsilon}} + \delta' \ ,$$

for any $\delta' > 0$.

To calculate the closed-formed solution of ε and δ we first need to state the following lemma.

Lemma 4.12. *Let $a, b, c \geq 0$. The bound $ax\log(1+bx) \leq c$ is satisfied for all $0 \leq x \leq \sqrt{\frac{c}{ab}}$.*

Proof. Directly from $\log(1+bx) \leq bx$. □

Let $\tilde{\varepsilon} = e^{\varepsilon} - 1$, $\varepsilon' = (1+\tilde{\varepsilon})\tilde{\varepsilon}$, $c_1 = \frac{mk}{n}$, $c_2 = \sqrt{2m\log(\frac{1}{\delta'})}$ and $c_3 = \frac{k}{n}$. With this notation we have to look for ε' such that

$$\varepsilon^* \geq \left(c_1\varepsilon' + c_2\right)\log\left(1 + c_3\varepsilon'\right) \quad . \tag{4.11}$$

We conjecture a form $\varepsilon' = c_4\varepsilon^*$ and note that the inequality above will be satisfied if, for example (the $1/2$ constant here is arbitrary):

$$c_1c_4\varepsilon^*\log(1+c_3c_4\varepsilon^*) \leq \frac{\varepsilon*}{2} \quad , \tag{4.12}$$

$$c_2\log(1+c_3c_4\varepsilon^*) \leq \frac{\varepsilon^*}{2} \quad . \tag{4.13}$$

We recall that ε^* is the target privacy parameter, which is a user specified value. We will assume henceforth that $0 < \varepsilon^* \leq 1$. Thus, using Lemma 4.12 we have that c_4 must satisfy the following two bounds simultaneously:

$$c_4 \leq \sqrt{\frac{1}{2c_1c_3}} \Rightarrow c_4 \leq \frac{n}{k}\sqrt{\frac{1}{2m}} \quad , \tag{4.14}$$

$$c_4 \leq \frac{1}{2c_2c_4} \Rightarrow c_4 \leq \frac{n}{k\sqrt{8m\log(\frac{1}{\delta'})}} \quad . \tag{4.15}$$

By taking the smallest of the two upper bounds we get $c_4 = \frac{n}{k}\frac{1}{\sqrt{8m\log(\frac{1}{\delta'})}}$, which implies the choice

$$\varepsilon' = \frac{n}{k}\frac{\varepsilon^*}{\sqrt{8m\log(\frac{1}{\delta'})}} \quad . \tag{4.16}$$

Recalling that $\varepsilon = \log(1+\tilde{\varepsilon})$ and $\tilde{\varepsilon} = (-1+\sqrt{1+4\varepsilon'})/2$, we finally obtain

$$\varepsilon = \log\left(1 + \frac{-1+\sqrt{1+\frac{4n\varepsilon^*}{k\left(\sqrt{8m\log(1/\delta')}\right)}}}{2}\right) = \log\left(\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{n\varepsilon^*}{k\left(\sqrt{8m\log(1/\delta')}\right)}}\right) \quad . \tag{4.17}$$

For the sake of asymptotics we note that the choice above has the form $\varepsilon = O(\log(1 + n\varepsilon^*/k\sqrt{m}))$ (assuming that δ' is a constant).

Now for the δ parameter, we note that our choice of ε yields:

$$\delta = \frac{n(\delta^* - \delta')}{mke^{\varepsilon}} \tag{4.18}$$

$$= \frac{n(\delta^* - \delta')}{mk} \frac{1}{\frac{1}{2} + \sqrt{\frac{1}{4} + \varepsilon'}} \tag{4.19}$$

$$= \frac{n(\delta^* - \delta')}{mk} \frac{1}{\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{n}{k}\frac{\varepsilon^*}{\sqrt{8m\log(\frac{1}{\delta'})}}}} . \tag{4.20}$$

□

4.5 Average Sub-sampling Sensitivity of Linear Queries

Recall that a linear query A evaluated on $x = (x_1, \ldots, x_n) \in \mathbb{X}^n$ has the form $A(x) = (1/n)\sum_{i=1}^n A(x_i)$. To compute $\mathrm{AVSS}_2^{(n,k,\mathcal{D})}(A)$ we start by writing

$$\mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_{\tilde{x}\sim_k x}\left[\|A(x) - A(\tilde{x})\|_2^2\right]\right] =$$

$$\mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_{\tilde{x}\sim_k x}\left[\left\|\frac{1}{n}\sum_{i=1}^n A(x_i) - \frac{1}{k}\sum_{i=1}^k A(x_{\phi(i)})\right\|_2^2\right]\right] ,$$

where we used the notation $\tilde{x} = (x_{\phi(1)}, \ldots, x_{\phi(k)})$ for some random injective function $\phi : [k] \to [n]$. Now note that because the x_i are i.i.d. and $\tilde{x}$ is a random sub-sample we can assume without loss of generality that $\phi(i) = i$. Therefore, the expression above is equal to

$$\begin{aligned}
&\mathbb{E}_{x\sim\mathcal{D}^n}\left[\left\|\left(\frac{1}{n} - \frac{1}{k}\right)\sum_{i=1}^k A(x_i) + \frac{1}{n}\sum_{i=k+1}^n A(x_i)\right\|_2^2\right] \\
&= \left(\frac{1}{k} - \frac{1}{n}\right)^2 \mathbb{E}_{x\sim\mathcal{D}^n}\left[\left\|\sum_{i=1}^k A(x_i)\right\|_2^2\right] \\
&+ \frac{1}{n^2}\mathbb{E}_{x\sim\mathcal{D}^n}\left[\left\|\sum_{i=k+1}^n A(x_i)\right\|_2^2\right] \\
&- \frac{2}{n}\left(\frac{1}{k} - \frac{1}{n}\right)\sum_{i=1}^k \sum_{j=k+1}^n \mathbb{E}_{x_i,x_j\sim\mathcal{D}}\left[\langle A(x_i), A(x_j)\rangle\right] .
\end{aligned}$$

The last term can be simplified using the identity $\mathbb{E}_{x_i,x_j\sim\mathcal{D}}\left[\langle A(x_i), A(x_j)\rangle\right] = \|\mathbb{E}_{x\sim\mathcal{D}}\left[A(x)\right]_2^2\|$. Furthermore, the other two terms can be computed as follows:

$$\begin{aligned}
\mathbb{E}_{x\sim\mathcal{D}^n}\left[\left\|\sum_{i=1}^{k} A(x_i)\right\|_2^2\right] &= \sum_{i=1}^{k}\sum_{j=1}^{k}\mathbb{E}_{x_i,x_j\sim\mathcal{D}}\left[\langle A(x_i), A(x_j)\rangle\right] \\
&= k\mathbb{E}_{x\sim\mathcal{D}}\left[\|A(x)\|_2^2\right] + k(k-1)\|\mathbb{E}_{x\sim\mathcal{D}}\left[A(x)\right]_2^2\|, \\
\mathbb{E}_{x\sim\mathcal{D}^n}\left[\left\|\sum_{i=k+1}^{n} A(x_i)\right\|_2^2\right] &= (n-k)\mathbb{E}_{x\sim\mathcal{D}}\left[\|A(x)\|_2^2 + (n-k)(n-k-1)\|\mathbb{E}_{x\sim\mathcal{D}}\left[A(x)\right]_2^2\|\right].
\end{aligned}$$

By writing $\mu := \mathbb{E}_{x\sim\mathcal{D}}\left[A(x)\right]$ and $\Sigma := \mathbb{E}_{x\sim\mathcal{D}}\left[A(x)A(x)^\top\right]$, we get $\|\mathbb{E}_{x\sim\mathcal{D}}\left[A(x)\right]_2^2\| = \|\mu\|_2^2$ and $\mathbb{E}_{x\sim\mathcal{D}}\left[\|A(x)\|_2^2\right] = \mathrm{Tr}(\Sigma)$. Therefore, using a simple calculation we see that

$$\begin{aligned}
\mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_{\tilde{x}\sim_k x}\left[\|A(x) - A(\tilde{x})\|_2^2\right]\right] &= \left(\frac{1}{k} - \frac{1}{n}\right)^2\left(k\,\mathrm{Tr}(\Sigma) + k(k-1)\|\mu\|_2^2\right) \\
&\quad + \frac{1}{n^2}\left((n-k)\,\mathrm{Tr}(\Sigma) + (n-k)(n-k-1)\|\mu\|_2^2\right) \\
&\quad - \frac{2}{n}\left(\frac{1}{k} - \frac{1}{n}\right)k(n-k)\|\mu\|_2^2 \\
&= \left(\frac{1}{k} - \frac{1}{n}\right)\left(\mathrm{Tr}(\Sigma) - \|\mu\|_2^2\right) \ .
\end{aligned}$$

From the calculation above we see that the rate of $\mathrm{AVSS}_2^{(n,k,\mathcal{D})}(A) = \Theta(1/km)$, since (assuming that both $\|\mu\|$ and $\mathrm{Tr}(\Sigma)$ are constant with n):

$$\frac{1}{m}\left(\frac{1}{k} - \frac{1}{n}\right) = \frac{1}{km}\left(1 - \frac{k}{n}\right) = \Theta\left(\frac{1}{km}\right) \ , \tag{4.21}$$

where we used that $k \leq n$ implies $(1 - k/n) = \Theta(1)$.

Now we move on to the average utility analysis of sub-sample and average mechanism.

4.6 Average Utility of Sub-sample and Average Gaussian Mechanism: Average Case

In this section we show the connection between average utility of Gaussian mechanism and the average utility of sub-sample-and-average mechanism.

The expectation of *subsample-and-average* mechanism $\tilde{M}$ is presented in the following lemma.

Lemma 4.13. *Let $\tilde{M}^{(m,k)} : \mathbb{X}^n \to \mathbb{R}^d$ be a* sub-sample-and-average mechanism*, we have,*

$$\mathbb{E}\left[\tilde{M}^{(m,k)}(x)\right] = \mathbb{E}_{\tilde{x}\sim_k x}\left[A(\tilde{x})\right]. \tag{4.22}$$

Proof. Note that taking an expectation with respect to this mechanism amounts to taking expectations over the sub-samples $\tilde{x}_i$ and the multiple instantiations of $\tilde{M}$. Thus, we will write

$$\begin{aligned}\mathbb{E}\left[\tilde{M}^{(m,k)}(x)\right] &= \mathbb{E}_{\{\tilde{x}_i\}}\left[\mathbb{E}_{\tilde{M}}\left[\frac{1}{m}\sum_{i=1}^{m}\tilde{M}(\tilde{x}_i)\right]\right] \\ &= \mathbb{E}_{\tilde{x}\sim_k x}\left[\mathbb{E}_{\tilde{M}}\left[\tilde{M}(\tilde{x})\right]\right] \\ &= \mathbb{E}_{\tilde{x}\sim_k x}\left[A(\tilde{x})\right] \ ,\end{aligned}$$

where the last equality uses the fact that $\tilde{M}$ is an output perturbation mechanism. □

Theorem 4.14. *The utility of the sub-sample-and-average mechanism $\tilde{M}^{(m,k)}$ is,*

$$\mathrm{UT}_2^{(n,\mathcal{D})}(\tilde{M}^{(m,k)}) = \frac{1}{m}\mathrm{UT}_2^{(k,\mathcal{D})}(\tilde{M}) + \frac{1}{m}\mathrm{AVSS}_2^{(n,k,\mathcal{D})}(A)^2 + \frac{m-1}{m}\mathrm{ASB}_2^{(n,k,\mathcal{D})}(A)^2 \ . \tag{4.23}$$

Proof. The utility of $\tilde{M}^{(m,k)}$ can now be computed as

$$\begin{aligned}&\mathrm{UT}_2^{(n,\mathcal{D})}(\tilde{M}^{(m,k)}) = \mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}\left[\|A(x)-\tilde{M}^{(m,k)}(x)\|_2^2\right]\right] \\ &= \mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_{\{\tilde{x}_i\}}\left[\mathbb{E}_{\tilde{M}}\left[\left\|\left(A(x)-\frac{1}{m}\sum_{i=1}^{m}A(\tilde{x}_i)\right)+\left(\frac{1}{m}\sum_{i=1}^{m}A(\tilde{x}_i)-\tilde{M}^{(m,k)}(x)\right)\right\|_2^2\right]\right]\right] \\ &= \frac{1}{m^2}\mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_{\{\tilde{x}_i\}}\left[\mathbb{E}_{\tilde{M}}\left[\left\|\sum_{i=1}^{m}(A(x)-A(\tilde{x}_i))\right\|_2^2\right]\right]\right] \\ &+ \frac{1}{m^2}\mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_{\{\tilde{x}_i\}}\left[\mathbb{E}_{\tilde{M}}\left[\left\|\sum_{i=1}^{m}(A(\tilde{x}_i)-\tilde{M}(\tilde{x}_i))\right\|_2^2\right]\right]\right] \\ &+ \frac{2}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_{\{\tilde{x}_i\}}\left[\mathbb{E}_{\tilde{M}}\left[\langle A(x)-A(\tilde{x}_i), A(\tilde{x}_j)-\tilde{M}(\tilde{x}_j)\rangle\right]\right]\right] \ .\end{aligned}$$

We can now analyze each of the terms in this expression independently. The first term is closely related to the average random sensitivity of A. We can see that by noting that the expectation over $\tilde{M}$ has no effect in this term, and the expectation over the $\tilde{x}_i$ can be computed as follows:

$$\begin{aligned}
&\mathbb{E}_{\{\tilde{x}_i\}}\left[\left\|\sum_{i=1}^{m}(A(x)-A(\tilde{x}_i))\right\|_2^2\right] \\
&=\sum_{i=1}^{m}\sum_{j=1}^{m}\mathbb{E}_{\tilde{x}_i}\left[\mathbb{E}_{\tilde{x}_j}\left[\langle A(x)-A(\tilde{x}_i),A(x)-A(\tilde{x}_j)\rangle\right]\right] \\
&=m\mathbb{E}_{\tilde{x}}\left[\|A(x)-A(\tilde{x})\|_2^2+m(m-1)\|A(x)-\mathbb{E}_{\tilde{x}}\left[A(\tilde{x})\right]\|_2^2\right] .
\end{aligned}$$

Thus, the first term in utility of $\tilde{M}^{(m,k)}$ can be written as

$$\begin{aligned}
&\frac{1}{m^2}\mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_{\{\tilde{x}_i\}}\left[\mathbb{E}_{\tilde{M}}\left[\left\|\sum_{i=1}^{m}(A(x)-A(\tilde{x}_i))\right\|_2^2\right]\right]\right] \\
&=\frac{1}{m}\mathrm{AVSS}_2^{(n,k,\mathcal{D})}(A)^2+\frac{m-1}{m}\mathrm{ASB}_2^{(n,k,\mathcal{D})}(A)^2 .
\end{aligned}$$

The second term is related to the utility of mechanism $\tilde{M}$. We start by seeing using that $\tilde{M}$ is an output perturbation mechanism the first two inner expectations simplify to

$$\begin{aligned}
&\mathbb{E}_{\{\tilde{x}_i\}}\left[\mathbb{E}_{\tilde{M}}\left[\left\|\sum_{i=1}^{m}\left(A(\tilde{x}_i)-\tilde{M}(\tilde{x}_i)\right)\right\|_2^2\right]\right] \\
&=\mathbb{E}_{\{\tilde{x}_i\}}\left[\sum_{i=1}^{m}\sum_{j=1}^{m}\mathbb{E}_{\tilde{M}}\left[\langle A(\tilde{x}_i)-\tilde{M}(\tilde{x}_i),A(\tilde{x}_j)-\tilde{M}(\tilde{x}_j)\rangle\right]\right] \\
&=m\mathbb{E}_{\tilde{x}\sim_k x}\left[\mathbb{E}_{\tilde{M}}\left[\|A(\tilde{x})-\tilde{M}(\tilde{x})\|_2^2\right]\right]
\end{aligned}$$

Now note that taking the expectation $\mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_{\tilde{x}\sim_k x}\left[.\right]\right]$ over a function that only depends on $\tilde{x}$ is equivalent to taking the expectation $\mathbb{E}_{\tilde{x}\sim\mathcal{D}^k}\left[.\right]$ over the same function. Therefore, the second term in the utility of $\tilde{M}^{(m,k)}$ is equal to

$$\begin{aligned}
&\frac{1}{m^2}\mathbb{E}_{x\sim\mathcal{D}^n}\left[\mathbb{E}_{\{\tilde{x}_i\}}\left[\mathbb{E}_{\tilde{M}}\left[\left\|\sum_{i=1}^{m}\left(A(\tilde{x}_i)-\tilde{M}(\tilde{x}_i)\right)\right\|_2^2\right]\right]\right] \\
&=\frac{1}{m}\mathbb{E}_{\tilde{x}\sim\mathcal{D}^k}\left[\mathbb{E}_{\tilde{M}}\left[\|A(\tilde{x})-\tilde{M}(\tilde{x})\|_2^2\right]\right] \\
&=\frac{1}{m}\mathrm{UT}_2^{(k,\mathcal{D})}(\tilde{M}) .
\end{aligned}$$

Finally, the third term can be shown to be equal to zero by observing that because $\tilde{M}$ is an output perturbation we have

$$\mathbb{E}_{\tilde{M}}\left[\langle A(x)-A(\tilde{x}_i),A(\tilde{x}_j)-\tilde{M}(\tilde{x}_j)\rangle\right]=\langle A(x)-A(\tilde{x}_i),A(\tilde{x}_j)-\mathbb{E}_{\tilde{M}}\left[\tilde{M}(\tilde{x}_j)\rangle\right]=0 . \tag{4.24}$$

In conclusion, the utility of the subs-ample-and-average mechanism for $\tilde{M}$ can be written as follows:

$$\mathrm{UT}_2^{(n,\mathcal{D})}(\tilde{M}^{(m,k)}) = \frac{1}{m}\mathrm{UT}_2^{(k,\mathcal{D})}(\tilde{M}) + \frac{1}{m}\mathrm{AVSS}_2^{(n,k,\mathcal{D})}(A)^2 + \frac{m-1}{m}\mathrm{ASB}_2^{(n,k,\mathcal{D})}(A)^2 \ . \quad (4.25)$$

□

The following corollary extends the results of Theorem 4.14 for the case of linear queries.

Corollary 4.15. *Suppose $A : \mathbb{X}^n \to \mathbb{R}^d$ is a linear query. Let M be the (ε, δ)-DP Gaussian mechanism for A and $\tilde{M}^{(m,k)}$ the sub-sample-and-average mechanism for A. We then have,*

$$\mathrm{UT}_2^{(n,\mathcal{D})}(\tilde{M}^{(m,k)}) = \frac{2\log(1.25/\tilde{\delta})}{m}\frac{d\mathrm{GS}_2^{(1)}(A)^2\log(1/\tilde{\delta})}{k^2\tilde{\varepsilon}^2} + \frac{1}{m}\mathrm{AVSS}_2^{(n,k,\mathcal{D})}(A)^2 \ . \quad (4.26)$$

Proof. Recalling that the average sub-sampling bias of linear queries is zero, we see that in the case where $\tilde{M}$ is an output mechanism for a linear query A we can further simplify the equation (4.25) to:

$$\mathrm{UT}_2^{(n,\mathcal{D})}(\tilde{M}^{(m,k)}) = \frac{1}{m}\mathrm{UT}_2^{(k,\mathcal{D})}(\tilde{M}) + \frac{1}{m}\mathrm{AVSS}_2^{(n,k,\mathcal{D})}(A)^2 \ . \quad (4.27)$$

If we further assume that $\tilde{M}$ is an $(\tilde{\varepsilon}, \tilde{\delta})$-DP Gaussian mechanism for a linear query A, then the utility formula from theorem 4.9 yields:

$$\mathrm{UT}_2^{(n,\mathcal{D})}(\tilde{M}^{(m,k)}) = \frac{2\log(1.25/\tilde{\delta})}{m}\frac{d\mathrm{GS}_2^{(1)}(A)^2\log(1/\tilde{\delta})}{k^2\tilde{\varepsilon}^2} + \frac{1}{m}\mathrm{AVSS}_2^{(n,k,\mathcal{D})}(A)^2 \ . \quad (4.28)$$

□

We note here that the second term will have rate $o(n^{-2})$ as long as $\mathrm{AVSS}_2^{(n,k,\mathcal{D})}(A) = o(\sqrt{m}/n)$. To continue with the utility analysis, we first need to adjust the privacy parameters with respect to the choice of sub-sampling parameters, sub-sample size k and number of sub-samples m. Thus, in the following, we provide the privacy analysis of sub-sample-and-average Gaussian mechanism, and then we continue our utility analysis in the future sections of this chapter.

We summarize the result of this section in the following theorem, which demonstrates the dependency between the rate assumption for the average sub-sampling sensitivity of A and the average utility of sub-sample-and-average mechanism.

Theorem 4.16. *Suppose $A : \mathbb{X}^n \to \mathbb{R}^d$ is a linear query. Let M be the (ε, δ)-DP Gaussian mechanism for A and $\tilde{M}^{(m,k)}$ be the subsample-and-average (ε, δ)-DP Gaussian mechanism for A. Suppose $k = \omega(1)$ and $m = \omega(1)$ are functions of n satisfying $k^2m = \omega(n^2)$ and $k^2m = \omega(n^2\log(km/n))$. If $\mathcal{D}$ is a distribution over $\mathbb{X}$ such that $\mathrm{AVSS}_2^{(n,k,\mathcal{D})} = o(\sqrt{m}/n)$, then $\mathrm{UT}_2^{(n,\mathcal{D})}(\tilde{M}^{(m,k)}) = o(\mathrm{UT}_2^{(n,\mathcal{D})}(M))$.*

Proof. Now we can use the expressions for $\tilde{\varepsilon}$ and $\tilde{\delta}$ obtained in Theorem 4.11 to estimate the first term in $\mathrm{UT}_2^{(n,\mathcal{D})}(\tilde{M})$. By simply plugging the two formulas into the first term of equation (4.28) we get (ignoring the constant term $Cd\mathrm{GS}_2^{(1)}(A)^2$):

$$\begin{aligned}
\frac{\log(1/\tilde{\delta})}{mk^2\tilde{\varepsilon}^2} &= \frac{\log\left(\frac{mk}{n(\delta-\delta')}\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right)\right)}{mk^2\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right)^2} \\
&= \frac{\log\left(\frac{mk}{n(\delta-\delta')}\right)+\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right)}{mk^2\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right)^2} \\
&= \frac{\log\left(\frac{mk}{n(\delta-\delta')}\right)}{mk^2\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right)^2}+\frac{1}{mk^2\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right)}
\end{aligned}$$

Since k and m can be chosen, suppose we have $k=\omega(1)$ and $m=\omega(1)$ with $k^2m=\omega(n^2)$, or equivalently $n=o(k\sqrt{m})$. Then $n/k\sqrt{m}=o(1)$ and we have

$$\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right)=\log\left(1+\sqrt{1+o(1)}\right)-\log(2)=o(1)\ .$$

To get more precise information about the behavior of this function note that for $x\approx 0$ a Taylor expansion yields

$$\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+x}\right)=x-\frac{3x^2}{2}+O(x^3)\ . \tag{4.29}$$

Before plugging $x=n\varepsilon/k\sqrt{m}\sqrt{8\log(1/\delta')}$ into the expression above we first note that $k\sqrt{m}=\omega(n)$ implies

$$\begin{aligned}
k^2mx &= \Theta\left(nk\sqrt{m}\right)=\omega(n^2)\ , \\
k^2mx^2 &= \Theta(n^2)\ , \\
k^2mx^3 &= \Theta\left(\frac{n^3}{k\sqrt{m}}\right)=o(n^2)\ .
\end{aligned}$$

Therefore we see that under the assumption on the rate of $k\sqrt{m}$ we obtain that the logarithmic term above satisfies:

$$k^2 m \log\left(\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right) = \frac{nk\sqrt{m}\varepsilon}{\sqrt{8\log(1/\delta')}} - \frac{3n^2\varepsilon^2}{16\log(1/\delta')} + o(n^2)$$
$$= \frac{n^2\varepsilon}{\sqrt{8\log(1/\delta')}}\left(\frac{k\sqrt{m}}{n} - \frac{3\varepsilon}{\sqrt{32\log(1/\delta')}}\right) + o(n^2).$$

In particular, since $k\sqrt{m}/n = \omega(1)$, this function is $\omega(n^2)$, which implies that the second term in our expression for $\log(1/\tilde{\delta})/mk^2\tilde{\varepsilon}^2$ is of the form $o(n^{-2})$.

To deal with the first term in our expression for $\log(1/\tilde{\delta})/mk^2\tilde{\varepsilon}^2$ we need to consider a Taylor expansion of the square of the logarithm above when $x \approx 0$, which will have the form:

$$\log\left(\frac{1}{2} + \sqrt{\frac{1}{4} + x}\right) = x^2 + O(x^3) \ . \qquad (4.30)$$

This implies that

$$\frac{\log\left(\frac{mk}{n(\delta-\delta')}\right)}{mk^2\log\left(\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right)^2} = \frac{\log\left(\frac{mk}{n}\right) + \log\left(\frac{1}{\delta-\delta'}\right)}{\frac{n^2\varepsilon^2}{8\log(1/\delta')} + o(n^2)}$$
$$= \Omega\left(\frac{\log\left(\frac{mk}{n}\right)}{n^2}\right) \ .$$

Plugging this observation into the expression above we now get

$$\frac{\log(1/\tilde{\delta})}{mk^2\tilde{\varepsilon}^2} = \Omega\left(\frac{8\log(1/\delta')\left(\log(mk/n) + \log(1/(\delta-\delta'))\right)}{n^2\varepsilon^2}\right) \ .$$

This expression will have order $o(n^{-2})$ (and therefore beat the utility rate of the Gaussian mechanism) as long as $\log(mk/n)/mk^2 = o(n^{-2})$, or equivalently $k^2 m = \omega(n^2\log(km/n))$. $\square$

4.7 Average Utility of Sub-sample and Average Gaussian Mechanism: Extreme Case

Consider the analysis in the previous section we suppose $k\sqrt{m} = n$. With this choice we see immediately that:

$$\begin{aligned}
\frac{\log(1/\tilde{\delta})}{mk^2\tilde{\varepsilon}^2} &= \frac{\log\left(\frac{mk}{n(\delta-\delta')}\right)}{mk^2\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right)^2} + \frac{1}{mk^2\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{n\varepsilon}{k\sqrt{8m\log(1/\delta')}}}\right)} \\
&= \frac{\log\left(\frac{mk}{n}\right)+\log\left(\frac{1}{\delta-\delta'}\right)}{mk^2\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{\varepsilon}{\sqrt{8\log(1/\delta')}}}\right)^2} + \frac{1}{mk^2\log\left(\frac{1}{2}+\sqrt{\frac{1}{4}+\frac{\varepsilon}{\sqrt{8\log(1/\delta')}}}\right)} \\
&= \tilde{\Theta}\left(\frac{\log\left(\frac{mk}{n}\right)}{mk^2\varepsilon^2}\right) ,
\end{aligned}$$

where we have hidden the dependence on δ and δ' inside the $\tilde{\Theta}$ for convenience. Therefore we see that ignoring the term $\log(mk/n)$, we get the same rate as with the assumption $k\sqrt{m} = \omega(n)$. Furthermore, since the term $\log(mk/n)$ is increasing with mk, we would like to make this as small as possible, which will be attained for a choice satisfying $k\sqrt{m} = n$ (among the ones that give overall rate $\tilde{O}(n^{-2})$). In particular, if there were no constraints to be satisfied, the optimal rate would be achieved (not surprisingly!) for $k = n$ and $m = 1$. However, the application of the subsampling lemma requires $k \leq n/2$, so all we can do is take $k = n/2$ and $m = 4$. In any case, for this choice (assuming that δ' is a small constant factor of δ) we get

$$\frac{\log(1/\tilde{\delta})}{mk^2\tilde{\varepsilon}^2} = \Theta\left(\frac{\log(1/\delta)^2}{n^2\varepsilon^2}\right) ,$$

which is only worse than the rate of the Gaussian mechanism by a factor $\log(1/\delta)$.

Aggregation of the results obtained in Sections 4.5, 4.6 and 4.7 we conclude the utility analysis of our proposed sub-sample and average framework in the following theorem.

Theorem 4.17. *Suppose $A : \mathbb{X}^n \to \mathbb{R}^d$ is a linear query. Let $\tilde{M}^{(m,k)}$ be the sub-sample-and-average (ε,δ)-DP Gaussian mechanism for A. Suppose $k = \omega(1)$ and $m = \omega(1)$ are functions of n satisfying $mk^2 = \Theta(n^2)$. If $\mathcal{D}$ is a distribution over $\mathbb{X}$, then*

$$\mathrm{UT}_2^{(n,\mathcal{D})}(\tilde{M}^{(m,k)}) = \Theta\left(\frac{\log\left(\frac{mk}{n}\right)}{mk^2\varepsilon^2} + \frac{1}{km^2}\right) . \tag{4.31}$$

For example, if we would like to make this rate $\Theta(n^{-2})$ we would need to take $km^2 = \Theta(n^2)$, which is achievable since it does not contradict the optimal choice $mk^2 = \omega(n^2)$ for the sub-sampling investigated above.

4.8 Algorithm and Experimental Evaluation

In this section, we conduct a series of experiments to illustrate the performance of our proposed sub-sampling framework against the baseline algorithms proposed in Chapter 3. We summarize the result of this chapter in Algorithm 3.

Algorithm 3 $\mathcal{A}$ (Sub-sample and average private policy evaluation)

```
Require: $X$, $\Phi$, $\gamma$, $R_{\max}$, $\varepsilon^*$, $\delta^*$, $\delta'$
  Let $\Theta = \{\}$
  Choose number of sub-sample parameter $m$ and sub-sample size $k$ that minimize eq.
  (4.31) such that they satisfy conditions of Theorem 4.17.
  Compute $\varepsilon$ and $\delta$ using equations (4.9) and (4.10) w.r.t. the choices of $m$ and $k$.
  do
      Let $u_i \leftarrow$ sub-sample $k$ data-points from $X$ uniformly at random WOR.
      Let $\theta_i \leftarrow \mathcal{M}(u_i, \Phi, \gamma, R_{\max}, \varepsilon, \delta, \delta')$.    ▷ $\mathcal{M} \in \{DP - LSL, DP - LSW\}$
      Add $\theta_i$ to $\Theta$.
      Let $\hat{\theta}_{\tilde{M}} \leftarrow \tilde{M}(\Theta)$.    ▷ cf. (4.5)
  while $(i < m)$
  Return $\hat{\theta}_g$.
```

4.8.1 Experiment Setup

For the sake of comparison with the results we presented previously, we use the same synthetic Markov chain domain as in Section 3.8 of Chapter 3. The chain has N states; with probability p, the agent stays in the same state, while with probability $(1 - p)$, it moves to wards the right. There is a reward of 1 when the agent reaches the final, absorbing state and 0 for all other states. With an initial state distribution (in our case, uniform over $\mathcal{S}$), a new trajectory is started and generating transitions are generated according to the described probabilities until the absorbing state is reached. A batch of such trajectories is sent as input to Algorithm 3. We conduct experiments in both the tabular and the function approximation settings. In the function approximation setting, we simply aggregate pairs of adjacent states, which are then forced to take the same value. We compared the proposed private algorithms with DP-LSW, DP-LSL and their

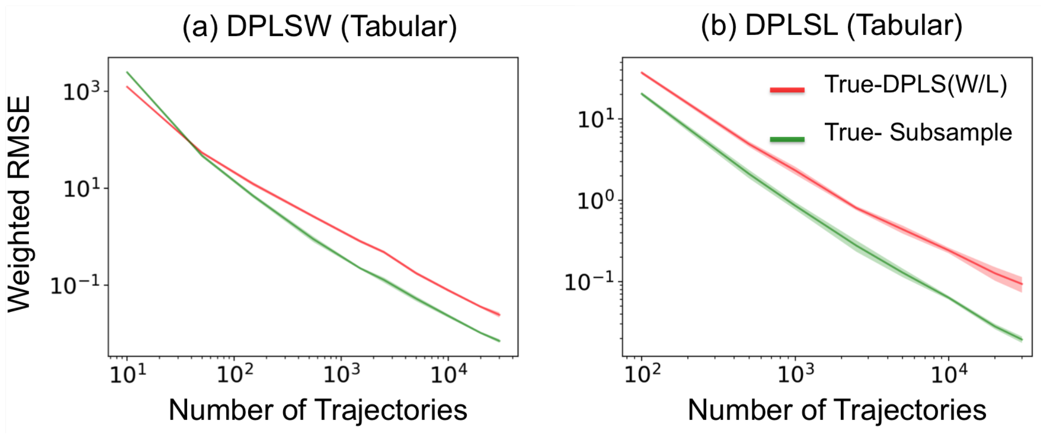


FIGURE 4.2: Empirical comparison between Sub-sampled DPLSW (DPLSL) and original DP-LSW (DPLSL) in the tabular setting. The green curve depicts the weighted RMSE based on the difference between the true value estimates and value estimates based on sub-sampling method we used in the chapter. The red curve demonstrates the weighted RMSE based on the difference between the true value estimates and the value estimates of original DPLSW (a) and DPLSL (b).

non-private equivalents LSW and LSL. The performance measure used is the average root mean squared error over the state space, evaluated with respect to the initial state distribution. The standard errors are computed over 5 independent runs. To show the consistency of Algorithm 3 in different privacy regimes, we also conduct a range of experiments for different values of ε.

The results are summarized in Figures 4.2, 4.3, 4.4 and 4.5, for an environment with $N = 40$ states, $p = 0.5$, discount $\gamma = 0.99$. For the differentially private algorithms, the privacy parameters are $\varepsilon \in \{0.1, 0.01, 0.001, 0.0001, 0.00001\}$ and $\delta = 0.1$. In general, these constants should be chosen depending on the privacy constraints of the domain. The sub-sampling parameters m (number of sub-samples) and k (sub-sample size) are chosen adaptively based on the theoretical results presented so far in this chapter.

Metric of success: we measure the performance of the Algorithm 3 in terms of *weighted root mean square error (RMSE)* of the state value vector, compared to the true value function (which in this case can be computed exactly by dynamic programminng), where the weight of each state is the same as initial state probability assigned to that state. The initial state distribution adopted for these experiments is the uniform distribution over $\mathcal{S}$.

The experiments show that the utility of both DP-LSW and DP-LSL is significantly improved by the use of sub-sampling. Our theoretical results clearly show the impact of the sub-sampling parameters on privacy-utility trade-offs, so we do not provide extensive experiments with different values.

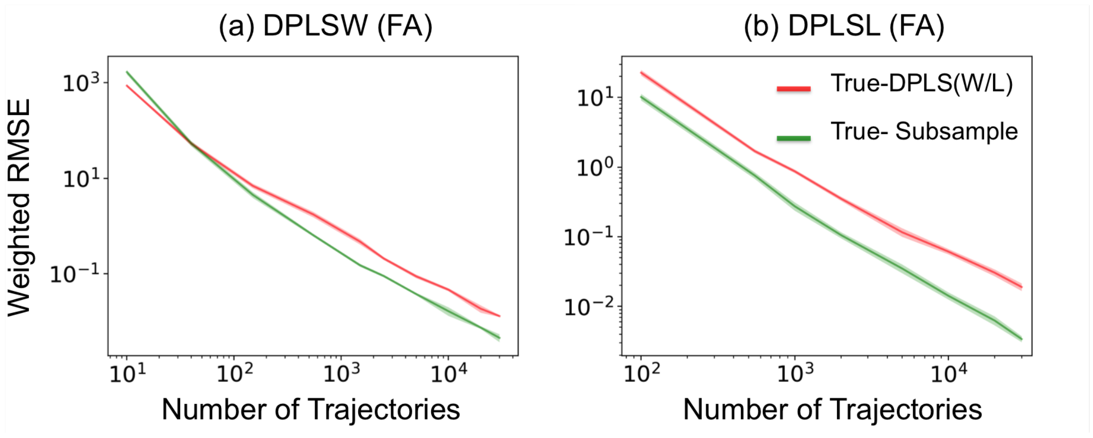


FIGURE 4.3: Empirical comparison between Sub-sampled DPLSW (DPLSL) and original DP-LSW (DPLSL) in the function approximation (FA) setting with the aggregation factor 2. The green curve depicts the weighted RMSE based on the difference between the true value estimates and value estimates based on sub-sampling method we used in the chapter. The red curve demonstrates the weighted RMSE based on the difference between the true value estimates and the value estimates of original DPLSW (a) and DPLSL (b).

The left plot in Figure 4.2 compares classic DP-LSW (the red curve) with Algorithm 3 that employs DP-LSW as its base mechanism (the green curve) and the sub-sample and average mechanism that employs DP-LSW mechanism after the aggregation (the blue curve) in the tabular setting. The right plot in Figure 4.2 demonstrates similar comparison in the same setting except that the base mechanism is DP-LSL. As the sample size increases Algorithm 3 for both DP-LSW and SP-LSL exhibits much faster convergence rate in comparison to the other two algorithms.

Similar experiments are conducted in the function approximation setting with the aggregation factor 2, where two consecutive states are assigned the same values. As can be seen in Figure 4.3, Algorithm 3 (green curve) still exhibits a very stable behaviour with both base algorithms and interestingly with faster converge rate in comparison with the tabular setting. This agrees with our observation in Chapter 3 and shows that function approximation provides better accuracy in relatively smaller batch-size setting. This agreed with the intuition that using the same data to estimate fewer parameters means the effect of each individual trajectory is already obscured by the function approximation. In medical applications, one expects to have many attributes measured about patients, and to need aggressive function approximation in order to provide generalization. This result tells us that differentially private algorithms should be favoured in this case as well.

To show the advantage of our proposed framework over original DPLSL and DPLSW we conduct of range of experiments for different privacy regimes. In Figure 4.4 demonstrates the consistency of subsampled DPLSW over a range of ε values, changing from low privacy regime to high privacy regime. As the graphs depict the performance margin between DPLSW and subsampled

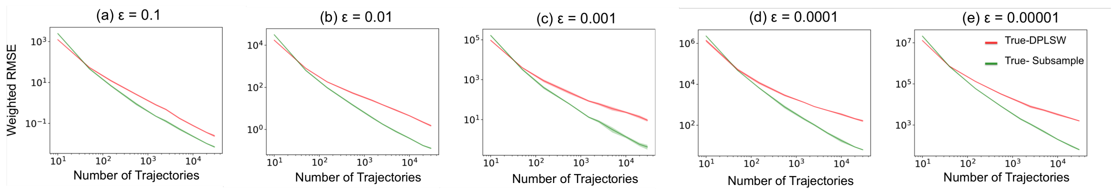


FIGURE 4.4: Empirical comparison of DPLSW(FA) and subsampled DPLSW for different values of ε from big (left) to small.

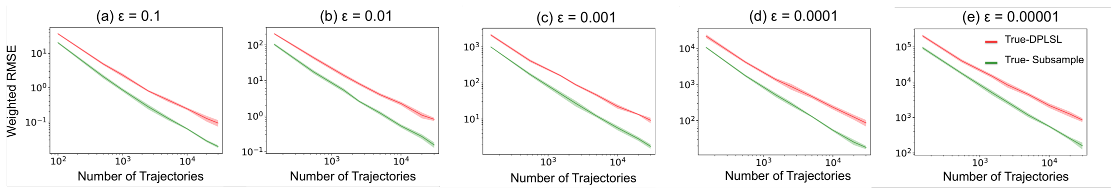


FIGURE 4.5: Empirical comparison of DPLSL(FA) and subsampled DPLSL for different values of ε, from big (left) to small.

DPLSW increases as we move towards higher privacy regimes. This phenomenon is less visible in the case of DPLSL (Figure 4.5), however we still observe that subsampled DPLSL maintains its significant performance margine as we move from low privacy to high privacy regime.

Overall, our experiments exhibits very promising results, showing that especially as batch size increases, our proposed sub-sampling mechanism greatly adapt with the new batch setting and the noise introduced by the DP mechanism decreases rapidly, and this adaptation to batch size significantly boosts the performance of baseline mechanisms.

4.9 Discussion

In this chapter, we have provided a systematic analysis of the utility advantage of using subsampling over classic diferentially private mechanisms in policy evaluation. In order to improve upon the utility of the DP-LSW and DP-LSL mechanisms introduced in the previous chapter, while maintaining sufficient privacy, we augmented these algorithms with a sub-sampling subroutine that dynamically adapts with the input batch mechanism. We have shown both theoretically and empirically that this approach can provide a utility boost. We suspect that stronger theoretical results can be obtained for this setting, and we hope that the work we presented can inspire others to investigate subsample-and-aggregate methods of various types.

Chapter 5

Membership Inference Attacks Against Deep Reinforcement Learning

In the previous chapters, we addressed the challenges of designing privacy-preserving reinforcement learning algorithms from the algorithm designer's perspective. In this chapter, we take the perspective of someone who wishes to attack models resulting from such algorithms. We study the extent to which reinforcement learning is vulnerable to a popular family of privacy attacks, known as membership inference attack.

Membership inference attacks or tracing attacks are special type of adversarial attacks designed to identify whether a data point has been used in the training of a given model obtained through a machine learning approach (Shokri et al., 2017; Yeom et al., 2018; Dwork et al., 2017). Differential privacy, as a rigorous standard, can be used to protect machine learning algorithms during the training phase, by alleviating the impact of a single data point on the behavior of a trained machine learning model. Recently, many open source library such as the TensorFlow Privacy framework (Ten) and the PyTorch Opacus framework (Opa) have been introduced to data scientists to facilitate the employment of differential privacy in classic machine learning. One of the goals of these frameworks is to protect machine learning algorithms against membership inference attacks!(Humphries et al., 2020; Dwork et al., 2017). However, some recent studies show that differential privacy algorithms are not necessarily successful in this task (Humphries et al., 2020; Ying et al., 2020).

The goal of this chapter is to examine the potential of carrying out these types of attacks against deep RL models. As deep RL becomes increasingly powerful for learning complex control strategies, and its applications are starting to include domains such as health care (eg adaptive treatment design) and dialog systems, in which user data is used to train the model, it is quite important to understand whether membership attacks can create a problem for the resulting models. While this question have been investigated in supervised learning, the RL setup is quite different due to the fact that training is done on trajectories of states, actions and rewards, and correlations between the information at different time steps could potentially be exploited. The potential vulnerability of deep RL models to membership attacks has not been explored sufficiently at the moment. In particular, there have been no concrete adversarial attack strategies in the literature tailored for deep reinforcement learning algorithms. To address this gap, we propose an adversarial attack framework tailored for testing the vulnerability of deep RL algorithms to membership inference attacks. We design a series of experiments to investigate the impact of temporal correlation, which naturally exists in reinforcement learning training data, on the probability of information leakage. Furthermore, we study the differences in the performance of *collective* and *individual* membership attacks against deep reinforcement learning algorithms. Experimental results show that the proposed adversarial attack framework is surprisingly effective at inferring the data used during deep reinforcement training with an accuracy exceeding 84% in individual and 97% in collective mode on two different control tasks in OpenAI Gym, which raises serious privacy concerns in the deployment of models resulting from deep reinforcement learning. Moreover, we show that the learning state of a reinforcement learning algorithm significantly influences the level of the privacy breach.

5.1 Membership Inference Attack in Machine Learning

In machine learning, a membership inference attack (MIA) or tracing attack (Dwork et al., 2017; Shokri et al., 2017) is a form of adversarial attack that is designed to infer the presence of a particular data point x in the training set of a target model $\mathcal{M}$. Specifically, when the target model is trained on a collection of sensitive data membership inference attacks become threatening. In this situation, the presences or absence of a data point is valuable piece of information for the adversary or attacker. The central intuition in the design of MIAs is that publicly available trained models tend to exhibit higher confidence in their predictions on the individuals who participated in the training data. Consequently, the members of training sets are vulnerable to privacy threats. The main challenge for the adversary in implementing MIAs is to design a classifier compatible with the target model domain setting and decide whether a particular data point was part of the training set given the output of the trained target model.

Based on the formalism provided by Yeom et al. (2018) we formally describe the attack training procedure. Let X be the training set of size n generated by the data oracle $\mathcal{O}_{\text{data}}$ and used to train the target model $\mathcal{M}$. Based on this information, the adversary trains a membership inference attack classifier $\mathcal{A}$ against $\mathcal{M}$ in order to extract whether a particular data point x was used in training of the target model $\mathcal{M}$. We note that we only assume black-box access to the data oracle;

thus, the data distribution is not assumed to be common knowledge. Therefore, if $x \in X$, the experiment $Exp(\mathcal{M}, \mathcal{A}, x, n) = 1$ and if $x \notin X$ then $Exp(\mathcal{M}, \mathcal{A}, x, n) = 0$.

Based on the existing literature on MIAs designed against models trained in supervised and unsupervised setting (Rigaki and Garcia, 2020; Hu et al., 2021), attackers employ different MIA design strategies based on: i) the adversary's knowledge level of the parameters in the target model, ii) the adversary's knowledge level of the training data distribution.

In the *Label-only* strategy (Yeom et al., 2018; Choquette-Choo et al., 2021), the attacker only relies on model predictions and discards the model's confidence scores. In this technique, the attacker uses the generalization gap (the difference between the train and test accuracy) in the attack model as the main driver in inferring the membership of individuals used in training the target model. The label-only technique was first introduced in Yeom et al. (2018) and was subsequently extended by Choquette *et al.* Choquette-Choo et al. (2021) to show how the label-only technique can improve the existing attack baselines. In the general RL setting, however, the notion of label is not defined. Hence, the label-only technique cannot be applied.

The *shadow model* technique (Shokri et al., 2017) is known as an effective and practical approach for designing membership inference attack models. Shadow models are parallel local models trained on data sets often sampled from the same distribution as the underlying distribution of the private data. In this method, the adversary trains the models with complete knowledge of the training set. Thus, using the auxiliary membership information and the trained shadow models, the adversary can build a membership classifier that identifies whether an individual has participated in the training of similarly trained models.

In both label-only and shadow model techniques, the adversary should have access to the model output labels and the training data true labels. However, the sequential nature of the training and output data points and the temporal nature of model training make the design of membership inference attacks for RL models fundamentally different. Moreover, the presence of replay buffer as an inevitable part of off-policy deep RL models adds another level of complexity to the design of membership inference attacks, as this intermediate transformation phase adds a new source of noise to the input data from the attacker's perspective.

5.2 Membership Inference Attack in Deep Reinforcement Learning

The vulnerability of deep reinforcement learning models to privacy breaches has only begun to be explored in literature. The only study on the privacy of deep reinforcement learning models Pan et al. (2019) demonstrated the potential vulnerability of deep reinforcement learning models to privacy breaches by adversarially inferring floor plans in grid world navigation tasks as well as the transition dynamics of continuous control environments from the models themselves. However, to the best of our knowledge, there has been no study on the potential membership leakage of the data directly employed in training deep reinforcement learning (deep RL) agents, which is known as membership inference attacks. The potential success of such membership inference attacks can have serious security ramifications in the deployment of models resulting from deep RL.

One of the major challenges in the implementation of membership inference attacks in deep RL settings is the sequential and correlated nature of the deep RL data points. For instance, in this context, a data point may consist of hundreds of correlated components in the form of tuples, which all together form a single trajectory. A successful membership inference attack algorithm should be able to learn not only the relation between the training and output trajectories but also the correlation between the tuples within each data point. Another complication in this regard concerns the type of relationship existing between the training and prediction data points. As an example, in the text generation problems (*e.g.* machine translation or dialog generation systems, there is a direct (usually one-to-one) correspondence between the input and output sequential data points. On the other hand, in deep RL settings, batches of collected data are used for training the deep RL policy, whose output corresponds to every single data point in the training batches.

In deep RL algorithms the concept of label is not defined as it is in supervised or semi-supervised learning methods. Instead, during the learning phase, the deep RL agent receives reinforcement (aka rewards) from the environment as the outcome of the selected action. The deep RL agent uses the obtained rewards to learn the task and optimize its learning policy, which produces output data points (trajectories) based on the trained data in the prediction phase. The aforementioned factors lead to complications with regard to defining input-output pairs in the training of attack classifiers and subsequently establishing a meaningful relationship between the pair constituents.

To gain a better understanding of the problem, we provide a brief introduction to the basics of reinforcement learning (For more details, refer to section 2.2). A data point in reinforcement learning is a sequence of temporally correlated tuples (s_t, a_t, r_t, s_{t+1}) that denote the history of the reinforcement learning agent's interaction with the environment from time $t = 0$ to $t = T$. This sequence of tuples is often referred to as *trajectory*. At time t, the reinforcement learning agent is at state s_t, interacts with the environment by taking action a_t according to a policy π, and subsequently receives the reward r_t and moves to the state s_{t_1}. The dynamics of the environment used by the policy π is not public information; thus, the reinforcement learning agent has no prior knowledge of the underlying environment dynamics. To test the vulnerability of reinforcement learning methods to *membership inference attacks*, we use *batch off-policy reinforcement learning* setting, where the common practice is that an (unknown) *exploration policy* (behaviour policy) π_b collects the input batch (private data). The batch data is thereafter delivered to the reinforcement learning algorithm in the form of independent trajectories (Markov chains) to train and release the target policy.

5.3 Problem Statement

Deep reinforcement learning methods have a unique structural difference compared to deep supervised or unsupervised methods, *i.e.* learning based on temporal correlation between the tuples in each trajectory and partial reinforcements the model receives upon interaction with the underlying environment. Even though deep reinforcement learning models decorrelate input trajectories through the replay buffer mechanism, the inherent correlation between transition tuples still plays a significant role in the behaviour of the output policy. For instance, recent studies show that feature representations learned by deep reinforcement learning models are highly correlated Mavrin et al. (2019). Yet, considering that replay buffer obscures the correlation of input trajectories through the process of decorrelation before passing them to the models, two natural questions arise:

1. How much information (with regard to the training members) can an adversary extract from the output of a trained off-policy deep RL agent?
2. To what extent can an adversary benefit from feature correlations in the learned policy?

In this study, we present the first black-box membership inference attack against a deep reinforcement learning agent to address the two aforementioned questions. In our proposed adversarial attack framework, the target model is considered a black box; thus, the attacker does not have access to the internal structure of the off-policy reinforcement learning agent. In particular, the attacker can only send a query to the target model and receive the answer in the form of a trajectory τ_T^{out}.

Our proposed attack framework tests the vulnerability of a state-of-the-art off-policy deep reinforcement learning model to membership attacks in two modes: *individual* and *collective*. In the individual mode, the attacker's goal is to train a probabilistic model that infers the membership probability of a trajectory τ_T^{in} given the trained policy π_f and the initial state s_0. In other words, the goal is to train a probabilistic classifier that learns the following distribution,

$$\Pr[(S_0 = s_0, A_0, R_0, S_1, A_1, R_1, \dots, S_T, A_T, R_T)^{\text{in}} | \pi_f, s_0]. \tag{5.1}$$

On the other hand, in the collective mode, the attacker's target is to predict the membership probability of a collection of data points. We show that reinforcement learning models are more vulnerable to collective membership inference attacks as in this mode, the attack classifier has more access to the required information to infer the input data. Moreover, we assess the vulnerability of the RL algorithm to membership inference attacks with respect to the learning state of the algorithm. Our results show that the cumulative amount of reinforcement the RL agent obtains in the course of training the policy affects the level of its vulnerability to membership inference attacks. Finally, we compare the impact of training the attacker with correlated data with that in decorrelated data on the quality of learning in the attack classifier.

5.4 Related Work

Membership inference attacks were used for the first time against machine learning systems by Shokri *et al.* (Shokri et al., 2017). In the following years, extensive studies were performed on the application of MIAs against supervised (Shokri et al., 2017; Long et al., 2020; Yeom et al., 2018; Salem et al., 2019; Song and Mittal, 2021) and unsupervised (Hayes et al., 2019b; Hilprecht et al., 2019; Chen et al., 2020) machine learning models, surveyed comprehensively by Hu et al. (2021), and Rigaki and Garcia (2020). In this chapter, bcasue of the sequential nature of data points in reinforcement learning we only review the existing attack models against supervised and unsupervised models trained on sequential data.

MIAs have been studied in the context of text generation problems (Song and Shmatikov, 2019; Hisamoto et al., 2020), where the attacker's goal is to identify whether or not a specific sequence-to-sequence or sequence-to-word pair is part of the input training data of a machine translation engine, a dialog system or a sentimental recommendation system. The structure of machine learning algorithms with sequential data differs from that of classic classification tasks regarding the input type and the type of prediction they output. While inputs and outputs in standard classification problems have fixed sizes, they are chains of correlated elements with variable lengths in sequence generation tasks. These differences pose a fundamentally different approach in designing MIAs against sequence generation tasks. The knowledge of output space distribution is no longer valid for the attack classifier since the output length may vary from one model to another. To tackle this challenge, Song and Shmatikov (Song and Shmatikov, 2019) assume access to a probability distribution over output-space vocabularies. Authors in Song and Shmatikov (2019) split their proposed attack model into two phases of shadow model training and audit model training. In the shadow model training phase, the attacker trains multiple shadow models assuming that the attacker has access to a generative model that generates a sequence of vocabularies. In the audit training phase, the attacker uses the rank of the words produced by the target model instead of the output probability distribution. The main assumption here is that the gap between rank prediction is associated with the words that appeared in the training and test sets. In a similar line of research, authors in Hisamoto et al. (2020) address membership inference attack against sequence-to-sequence models in the setting where the adversary is agnostic to the word sequence distribution. However, in this work, the attacker is equipped with a generative model for different translation subcorpora, an alternative for output word sequence distribution.

Apart from the machine translation setting, membership inference attacks have been executed against aggregate location time-series (Pyrgelis et al., 2017, 2018, 2020). For the first time, authors in Pyrgelis et al. (2020) study the impact of different spatial-temporal factors that contribute to the vulnerability of time-series-based algorithms to membership inference attacks.

Models trained on sequential data have the following fundamental differences with RL algorithms,

1. while in the language setting is the input-output relation is well defined and deterministic, this type of relationship is defined through the trained policy, and each output sequence can be considered as the evidence for the entire input dataset.
2. RL agent follows an online/active learning paradigm, while Machine Translation follows a supervised or unsupervised learning paradigm; thus the notion of labels in RL is undefined.
3. RL agent directly learns from the temporal correlation, and this is why temporal correlation becomes critical from both the attacker's and the RL agent's perspectives.

Due to the above-mentioned fundamental differences, one requires a fundamentally different approach in designing MIAs against the RL algorithms. To the best of our knowledge, there is no prior work in the context of deep reinforcement learning that addresses the problem of membership inference at a microscopic level, where the attacker infers the membership of a particular data-point in the training set of deep reinforcement learning models (Hu et al., 2021; Rigaki and Garcia, 2020),.

For the sake of completeness, we briefly review the only existing privacy attack against reinforcement learning. Pan et al. (2019) proposed a black-box attack against deep reinforcement learning algorithms that centers around the over-fitting problem. More specifically, the proposed attack studies the effect of over-fitting on revealing information about the agent's training environment as well as the model parameters. Shadow model training proposed in Pan et al. (2019) aims to infer the transition model used to train the target policy from the set of candidate transition dynamics. The assumption of having access to a collection of transition dynamics is infeasible to many real-world reinforcement learning settings and less appealing to the industrial audience, where the concern is the privacy of individuals participating in model training.

5.5 Attack Framework

In this section, first, we explain the general problem setting and subsequently introduce our attack platform and our proposed method of data formatting for training the attack models.

In the proposed adversarial attack framework, we successfully conduct membership inference attacks against deep RL in a black-box setting, where only the model output is accessible to the external users. The deep RL model interacts with an environment whose distribution of initial states, state space $\mathcal{S}$ and action space $\mathcal{A}$ are assumed to be common knowledge, an assumption that is widely accepted in the RL community (Sutton, 1985; Vietri et al., 2020; Szepesvári, 2010). For instance, in clinical trials, the initial state could be a category of diseases, with respect to which the RL clinical model will choose to take the proper action and train its policy based on the outcome (rewards) it observes.

We propose an adversarial attack method for studying the vulnerability of the deep RL algorithm to MIA in a black-box setting, where the attacker's access to the model is limited to the output trajectories of the model trained on given input trajectories. Figure 5.1 depicts the general framework of our proposed black-box attack on deep RL algorithms.

The two important oracles that always accompany the end-to-end design of a black-box attack model in off-policy deep reinforcement learning are: i) data oracle $\mathcal{O}_{\text{data}}$ and ii) model trainer oracle $\mathcal{O}_{\text{train}}$. The data oracle interacts with the environment and returns a set of independent and identically distributed (i.i.d.) training trajectories (Markov chains) for the model trainer oracle $\mathcal{O}_{\text{train}}$ (see Figures 5.1 (a, b)). The data oracle is a black box, which is equipped with a set of unknown exploration policies. To train the target model, whose training input is of the adversary's interest, the data oracle is initialized privately (see Figure 5.1 (a)), leading to the generation of a batch of private training data points in the form of trajectories. The model trainer oracle is agnostic to the exploration policy used for the data collection. The training data batch is passed to the deep RL trainer oracle, and the resulting trained model is made publicly available for data query. Our experimental framework can adopt any of the existing off-policy batch deep reinforcement learning models as the deep RL trainer oracle. In this study, we choose to work with the state-of-the-art Batch-Constrained deep Q-learning (BCQ) (Fujimoto et al., 2019) model, which exhibits remarkable performance in complex control tasks.

We use the *shadow model* (Shokri et al., 2017) training technique to acquire the data needed for training the attack classifier. In particular, the attacker provides the deep RL trainer oracle with a set of non-private training trajectories through the data oracle (Figure 5.1 (b)). The output trajectories act as pieces of evidence for the input trajectories. The attacker subsequently queries output trajectories from the trained deep RL model and passes the training and output trajectories to the data formatter (Figure 5.1 (c)). In this step, the trajectories are augmented into pairs based

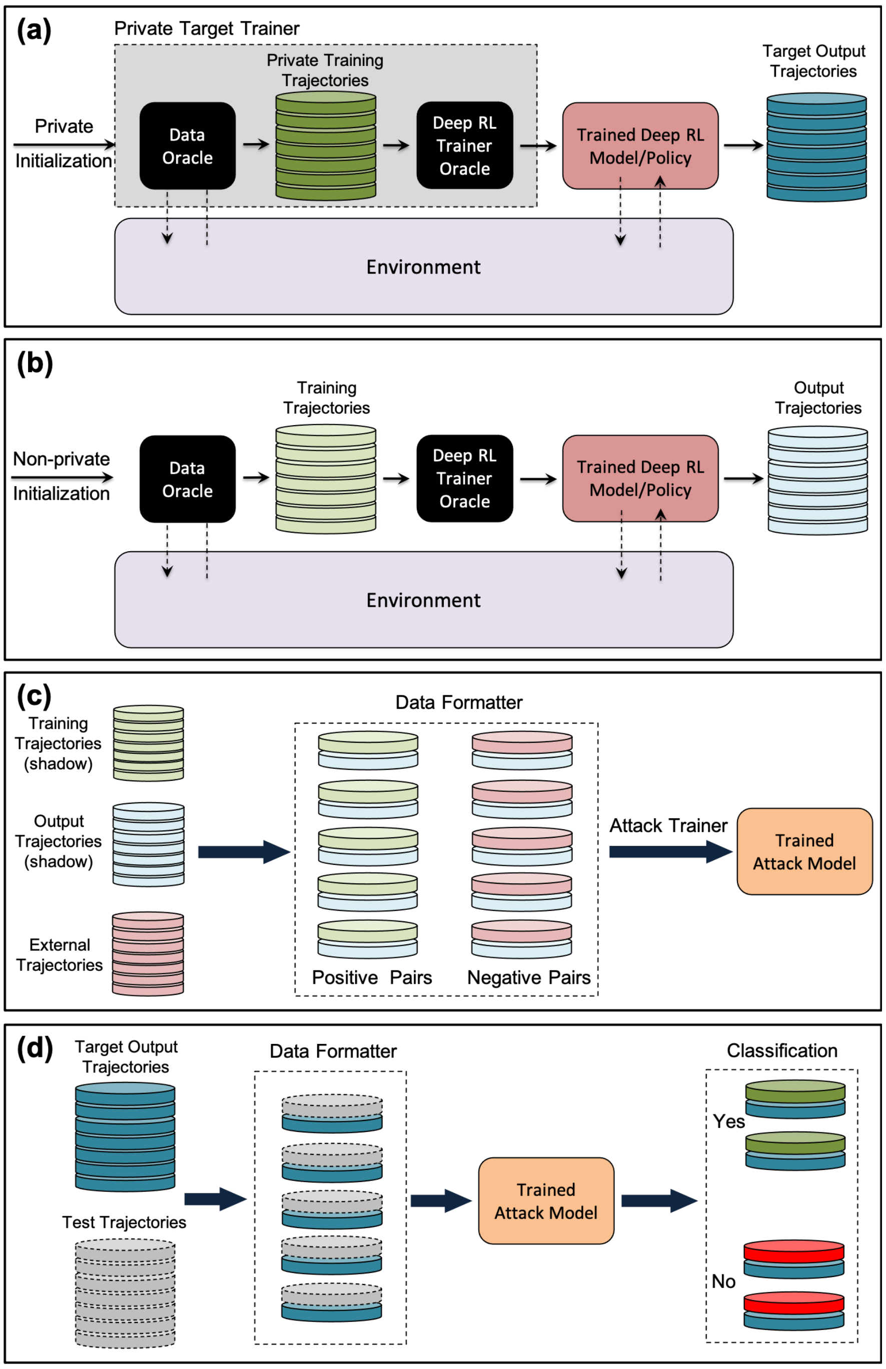


FIGURE 5.1: Proposed black-box membership inference attack architecture in deep reinforcement learning. **(a)** Private deep RL model training: the black-box exploration engine (data oracle) interacts with the environment and provides private training trajectories for the black-box deep RL model trainer. The trained deep RL model is subsequently used to output target trajectories through interaction with the environment. **(b)** Shadow training: the data and deep RL trainer oracles are used to produce the training and output trajectories in a non-private manner. **(c)** Training the attack classifier: the input and output trajectories obtained in part (b) are paired together in *data formatter* to provide positive training pairs for the attack model. Another set of trajectories, which has not been used in training the shadow model, is used with the output trajectories from part (b) to create negative training pairs for the attack model. The attack model is subsequently trained using the paired trajectories. **(d)** Membership inference attack: the target output trajectories are paired with sample test trajectories in the data formatter. The trained attack model subsequently uses the pairs to infer the test set trajectories that were used to train the private deep RL model.

on the internal logic of the attack trainer and are subsequently labelled. The training and output trajectories are labelled as positive or negative depending on whether or not the trajectories belong to the same trained model. Finally, the *attack trainer* trains a probabilistic classifier that takes as input the pairs of trajectories prepared by the data formatter and returns a trained attack classifier that is subsequently used to infer the membership of target input trajectories (Figures 5.1 (c,d)).

Since the attack training data collected by the data oracle $\mathcal{O}_{\text{data}}$ and prepared by the data formatter is of a sequential nature, we need to adopt an attack model that is compatible with time-series data. The classifier should minimize the expected loss, defined as

$$\mathbb{E}_{\mathcal{D}}\left[l(f(D,\pi_f),g(.))\right] \approx \frac{1}{|D|}\sum_{\tau \in D} l(f_{\boldsymbol{\theta}}(\tau,\pi_f),g(\tau,\pi_f)), \tag{5.2}$$

where $g(.)$ is the function that assigns labels to the formatted pairs, $f(.)$ is the parameterized classifier and $l(.)$ is the loss function adopted by f. The dataset D contains a set of i.i.d. trajectories drawn from $\mathcal{D}$ and π_f denotes the policy trained on D. The goal of the attacker is to train a classifier that learns a parameter vector (or network) θ^* that minimizes the loss function. We provide more details regarding the data formatter and the attack classifier in the following sections.

5.6 Experimental Setup

In this section, we mention the different settings we have considered in our experimental design. In our experimental design, we study the vulnerability of the deep RL model to membership inference attacks in terms of the following factors, including:

1) *the maximum trajectory length T_{max} within each episode* - The value of $T_{\max}$ is determined and fixed by the environment during data collection and model training. In particular, the RL agent's trajectory in each episode ends when either the agent arrives at an absorbing state at $T < T_{\max}$ or the number of time steps $T = T_{\max}$. Larger $T_{\max}$ corresponds to larger values of return (cumulative reward), thus improved deep RL policy.

2) *the membership inference mode (collective vs. individual MIA)* - In the individual mode, the adversary's goal is to infer the membership of *single* training data points (trajectories), while in the collective mode, the adversary's target is a *batch* of trajectories used in the training of the deep RL model.

3) *the level of correlation within the input trajectories used to train the attack classifier* - In the case of individual MIA, we study the performance of our proposed attack classifier in two modes: 1) *correlated* mode, where the adversary is trained on pairs with undisturbed input trajectories, 2) *decorrelated* mode, where the input trajectory is formed by sampling tuples at random from the whole batch. This set of experiments provides useful information regarding the effect of the correlation level within the input trajectories on the performance of the attack model.

A detailed description of the environments used in our experimental design, the data formatting technique, and the attack architecture is provided below.

5.6.1 Environments and RL Setting

We assess the the algorithms on OpenAI Gym environments (Brockman et al., 2016) powered by MuJoCo physics engine (Todorov et al., 2012), which are standard tasks adopted by many recent reinforcement learning studies (Lillicrap et al., 2015; Haarnoja et al., 2018; Fujimoto et al., 2018; Henderson et al., 2018; François-Lavet et al., 2018). OpenAI Gym provides a variety of simulated locomotion tasks with different action and state space dimensionalities. Here, we train the deep RL agent on two high-dimensional continuous control tasks: *Hopper-v2* ($\mathcal{A} \subset \mathbb{R}^3$ and $\mathcal{S} \subset \mathbb{R}^{11}$) *Half Cheetah-v2* ($\mathcal{A} \subset \mathbb{R}^6$ and $\mathcal{S} \subset \mathbb{R}^{17}$). Starting from virtually zero knowledge of how each task works, the deep RL agent's goal is to teach the robot in Hopper-v2 how to hop and the cheetah in HalfCheetah-v2 how to run.

We use the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2015) as the data oracle $\mathcal{O}_{\text{data}}$ and Batch-Constrained Deep Q-Learning (BCQ) Fujimoto et al. (2019) as the batch off-policy deep RL method used in the trainer oracle $\mathcal{O}_{\text{train}}$. The choice of data oracle is arbitrary; any exploration policy or data collection mechanism or subroutine that is compatible with batch off-policy deep RL setting is acceptable. For the set of experiments we perform in this study, we trained two DDPG models with distinct initial parameters to act as data oracle in order to provide the data required for the model training phase and two DDPG models with distinct initial parameters to act as data oracle in order to provide the data required for the model testing phase. Then, we trained one BCQ agent for the target modelling phase and one BCQ agent for the shadow modelling phase (see Figure 5.1).

In interaction with the MuJoCo environments, the data oracle $\mathcal{O}_{\text{data}}$ produces training trajectories for the trainer oracle $\mathcal{O}_{\text{train}}$, on which the deep RL agent is trained. The adversary uses the data and train oracles to train a shadow deep RL model and subsequently passes the training and model output trajectories to the data formatter. The next paragraph provides details regarding how the data formatter prepares the pairs for training the attack classifier.

5.6.2 Data Augmentation

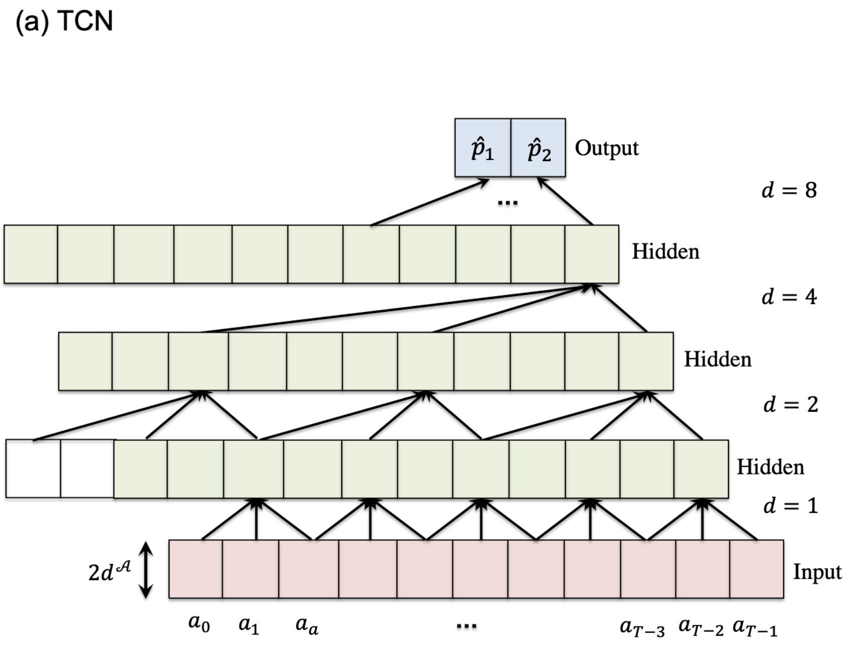
In episodic tasks, each trajectory starts with an initial state s_0 drawn from the available distribution of initial states in the environment, based on which the RL agent selects action a_0. The environment subsequently takes the agent to the next state s_1 and returns the reward r_1. The agent's next choice of action is based on s_1, and this cycle continues until the trajectory ends at s_T. In other words, the initial state s_0 plays a significant role in determining the sequence of taken actions by the RL policy and the consequent states and rewards. Thus, to prepare training pairs for the attack classifier, we pair training and output trajectories that have the same initial states. In this way, we fix the starting point of the two trajectories in a pair. Moreover, as the RL agent interacts with MDP, the resulting trajectory is a Markov chain. Thus, every state and reward in the trajectory is a consequence of the previous state and action. Therefore, we choose to remove states and rewards from the trajectories and use only the selected actions for preparing the pairs.

Each task is equipped with a set of absorbing states $\mathcal{B} \in \mathcal{S}$. The state that leads to the termination of an agent's chain of interactions in an environment is absorbing. Due to the presence of absorbing states in an environment, trajectories generated through the agent's interaction with the environment have different lengths. To pair the training and output action trajectories from the deep RL model, we need to either increase the length of shorter action trajectories to match that of the longest one, or clip longer action trajectories to a pre-determined length. Based on the desired length, we choose to repeat the last action in shorter action trajectories for the required number of times and trim longer trajectories. Note that setting the clipping length to large values in sparse tasks, where the trajectories often end at time steps much smaller than $T_{\max}$, is not desirable, as a considerable number of last-action repetitions in trajectories misleads the attack classifier.

Each action trajectory is a $d^{\mathcal{A}} \times T$ dimensional array, where $d^{\mathcal{A}}$ is the dimension of action space, and T is the total number of actions in the trajectory. The output action trajectory is concatenated with the RL training trajectory such that the resulting pair is a $2d^{\mathcal{A}} \times T$ dimensional array. The pairs are subsequently passed to the attack classifiers in multi-dimensional arrays $\mathbb{R}^{2d^{\mathcal{A}} \times T}$ and $\mathbb{R}^{2d^{\mathcal{A}} \times T \times m}$ in individual and collective modes, respectively. The value m refers to the number of pairs in each batch in the collective mode, which is set to $m = 50$ in this study.

5.6.3 Attack Classifier Architecture

We use Temporal Convolutional Networks (TCNs) Bai et al. (2018) as the classifier for individual MIA and Residual Network (ResNet) He et al. (2016) deep architecture for collective MIA. Figure 5.2 shows a schematic of TCN (Figure 5.2(a)) and ResNet (Figure 5.2(b)) network architectures.



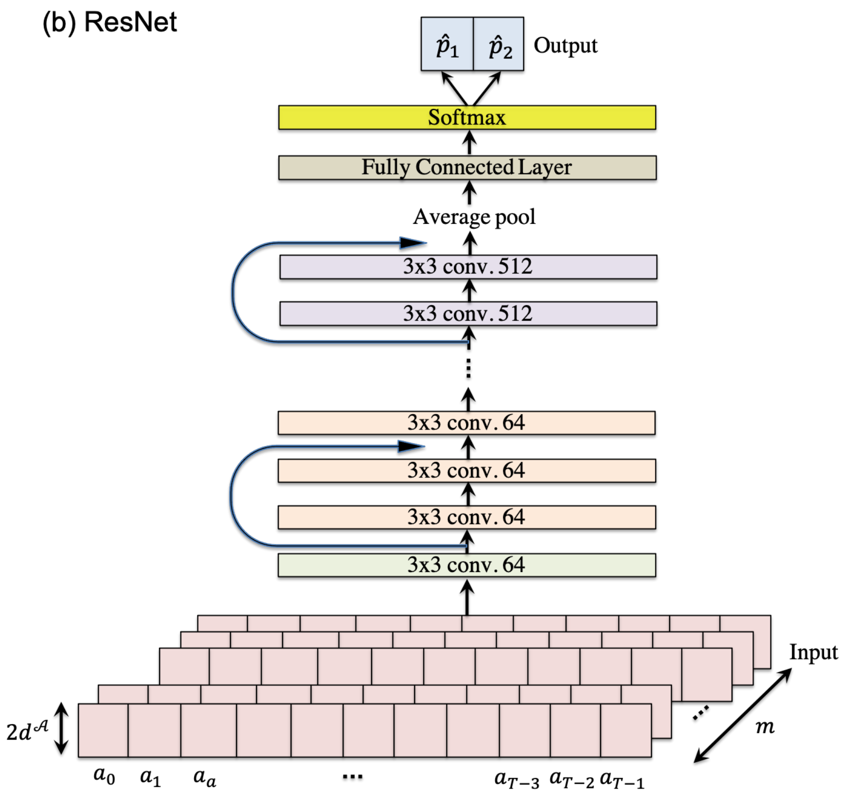


FIGURE 5.2: The network architecture of TCN **(a)** and ResNet **(b)** used in the individual and collective membership inference attacks, respectively.

5.6.3.1 Individual Mode Attack Classifier Architecture

In deep RL, both training and output trajectories are composed of temporally correlated transition tuples with variable lengths. Thus, the choice of attack classifier must capture the input level temporal correlation in its feature representation. TCNs are structurally designed to utilize the inherent temporal correlation in the training data through a hierarchy of temporal convolutions architecture. In this regard, TCN employs a 1D fully-convolutional network (FCN) architecture (Long et al., 2015), where each of its hidden layer has the same length as the input layer (Figure 5.2(a)). The main advantage of TCN is the ability to use dilation in convolution layers to keep the long-ranged temporal dependency and increase the receptive field of the convolutional layers. In the individual MIA mode, since the input data to the classifier is a collection of i.i.d. temporally correlated pairs (i.e. a two-dimensional tensor $\mathbb{R}^{2d^A \times T}$), the long-range correlation between input tuples within each trajectory is well-aligned with the input structure of TCNs. For more information on TCN architecture, please refer to Bai et al. (2018). Figure 5.2(a) demonstrates the general TCN architecture we used to design and train the individual mode attack classifier. TCN architecture we employed is composed of three hidden layers and one output layer that computes the membership probabilities. The input later for our proposed TCN architecture is a tensor $\mathbb{R}^{2d^A \times T}$ that is compatible with a pair of trajectories of length T.

5.6.3.2 Collective Mode Attack Classifier Architecture

In the collective mode, we choose to work with deep residual networks (ResNets) as the choice of attack classifier. We choose ResNet deep architecture because of its inherent compatibility with data sets with complex deep structures with temporal correlation within its elements (Zhang et al., 2017; Wen et al., 2018). In the collective mode, our input is in the form of three-dimensional tensor (*e.g.* $\mathbb{R}^{2d^A \times T \times m}$). Unlike the individual MIA mode, which involves 2-dimensional inputs, in the collective MIA mode, we have another dimension m for the number of trajectories in each batch of trajectories. Consequently, the collective MIA setting is similar to that in image classification (He et al., 2016; Simonyan and Zisserman, 2014; Huang et al., 2017) problems. Thus, we use ResNet (He et al., 2016) architecture, which is popular in solving standard computer vision problems (He et al., 2016; Minaee et al., 2021; Zhao et al., 2019). Our ResNet architecture is composed of 18 hidden layers with a a softmax output layer and three dimensional input tensor. Table 5.1 shows the deep network parameters used in this study in both networks.

TABLE 5.1: TCN and ResNet Architecture Settings

TCN Parameter	Value	ResNet Parameter	Value
Optimizer	Adam	*Optimizer*	Adam
Learning Rate	1e−3	*Learning Rate*	1e−3
Dropout value	0.5	*Weight decay*	1
Discount Factor	0.99	*Discount Factor*	0.99

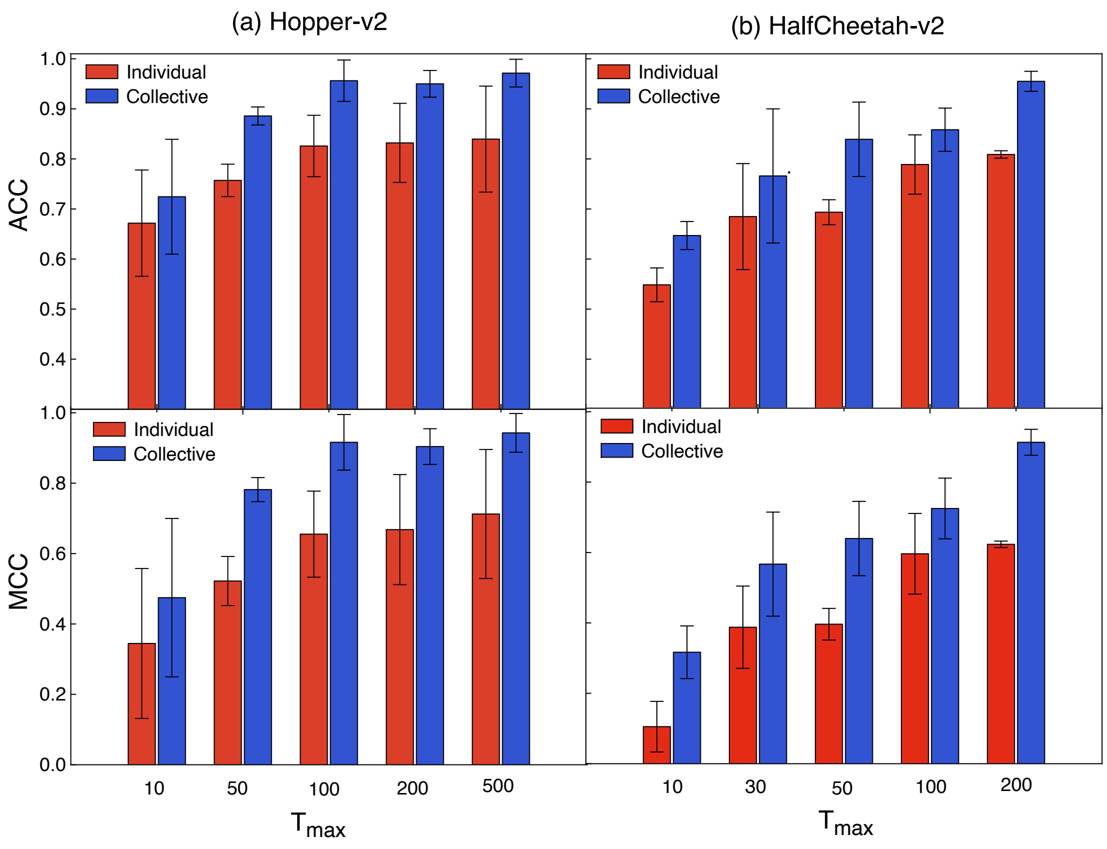


FIGURE 5.3: The performance of the attack classifiers in tasks Hopper-v2 **(a)** and HalfCheetah-v2 **(b)** in individual and collective attack modes. Each data point is determined from the average result of 5 separate runs. The error bars depict the error on the mean for ACC (*top*) and MCC (*bottom*) for the corresponding runs.

5.7 Results and Discussion

In this section, we discuss three different experimental scenarios to capture the interdependence between different parameters that affect the accuracy of membership attacks in deep reinforcement learning settings.

TABLE 5.2: The performance of the attack classifiers in Hopper-v2 **(a)** and HalfCheetah-v2 **(b)** for different maximum trajectory lengths $T_{\max}$ in terms of accuracy (ACC), precision (PR), recall (RE), F1 score (F1), and Matthews correlation coefficient (MCC). The values in parentheses show the results for the collective attack mode.

(a) Hopper-V2

T_{max}	ACC	PR	RE	F1	MCC
10	0.67 ± 0.11	0.73 ± 0.10	0.66 ± 0.10	0.70 ± 0.1	0.34 ± 0.21
	(0.72 ± 0.11)	(1 ± 0.00)	(0.65 ± 0.15)	(0.75 ± 0.12)	(0.47 ± 0.22)
50	0.76 ± 0.03	0.79 ± 0.06	0.73 ± 0.04	0.75 ± 0.03	0.52 ± 0.07
	(0.89 ± 0.02)	(0.94 ± 0.01)	(0.83 ± 0.04)	(0.88 ± 0.02)	(0.78 ± 0.03)
100	0.82 ± 0.06	0.84 ± 0.08	0.83 ± 0.03	0.83 ± 0.06	0.66 ± 0.12
	(0.96 ± 0.04)	(0.98 ± 0.02)	(0.93 ± 0.07)	(0.95 ± 0.05)	(0.92 ± 0.08)
200	0.83 ± 0.08	0.83 ± 0.09	0.87 ± 0.05	0.85 ± 0.07	0.67 ± 0.16
	(0.95 ± 0.03)	(0.96 ± 0.02)	(0.94 ± 0.05)	(0.95 ± 0.03)	(0.90 ± 0.05)
500	0.84 ± 0.11	0.92 ± 0.03	0.75 ± 0.23	0.78 ± 0.17	0.71 ± 0.18
	(0.97 ± 0.03)	(0.97 ± 0.03)	(0.98 ± 0.02)	(0.97 ± 0.03)	(0.94 ± 0.06)

(b) HalfCheetah-V2

T_{max}	ACC	PR	RE	F1	MCC
10	0.55 ± 0.03	0.58 ± 0.05	0.34 ± 0.06	0.43 ± 0.06	0.11 ± 0.07
	(0.65 ± 0.03)	(0.61 ± 0.01)	(0.80 ± 0.10)	(0.69 ± 0.04)	(0.32 ± 0.07)
30	0.68 ± 0.11	0.64 ± 0.09	0.85 ± 0.08	0.73 ± 0.08	0.39 ± 0.12
	(0.77 ± 0.13)	(0.82 ± 0.02)	(0.67 ± 0.32)	(0.70 ± 0.21)	(0.57 ± 0.14)
50	0.70 ± 0.02	0.71 ± 0.03	0.66 ± 0.09	0.67 ± 0.05	0.40 ± 0.04
	(0.84 ± 0.07)	(0.84 ± 0.08)	(0.84 ± 0.10)	(0.84 ± 0.08)	(0.64 ± 0.10)
100	0.79 ± 0.06	0.80 ± 0.06	0.78 ± 0.13	0.78 ± 0.08	0.60 ± 0.11
	(0.86 ± 0.04)	(0.89 ± 0.08)	(0.84 ± 0.02)	(0.86 ± 0.04)	(0.73 ± 0.09)
200	0.81 ± 0.01	0.79 ± 0.03	0.86 ± 0.04	0.82 ± 0.00	0.62 ± 0.01
	(0.96 ± 0.02)	(0.98 ± 0.01)	(0.93 ± 0.05)	(0.95 ± 0.02)	(0.91 ± 0.04)

5.7.1 Collective vs. Individual MIAs

We assess the behaviour of the attack classifiers in predicting the membership probability of each data point (individual attack mode) and collective data points (collective attack mode) using different classification metrics thoroughly explained in Chapter 2. Figure 5.3 presents the performance of the attack classifiers in the individual mode (TCN) and collective mode (ResNet) in Hopper-v2 (Figure 5.3(a)) and HalfCheetah (Figure 5.3(b)) environments in terms of ACC and MCC for different maximum trajectory lengths $T_{\max}$. The Full report of their performance in the two tasks is provided in Tables 5.2(a) and 5.2(b), respectively. The results show that our proposed attack framework is remarkably effective at inferring the RL model training data points. Considering that both classifiers are trained with only one shadow model, the obtained results demonstrate high privacy risks in employing deep reinforcement learning when working with sensitive training data.

Moreover, the results reveal that for a fixed $T_{\max}$, the adversary infers collective data points with significantly higher accuracy compared with that in the individual mode. For example, there are instances in the Hopper-v2 task, where the membership inference accuracy in the collective mode is more than 13% higher than that in the individual mode. This observation shows that the deep RL algorithm is more vulnerable to MIA in the collective mode, which is expected since more information is provided to the attack classifier through a batch of data points instead of one. Nevertheless, from the deep RL agent's perspective, specific information to each individual is concealed from the adversary in the collective mode, which is helpful in preserving the individuals' identities.

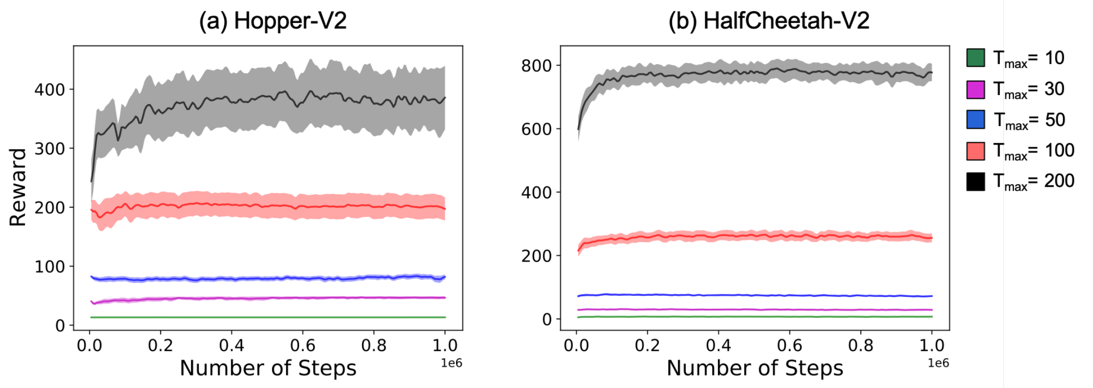


FIGURE 5.4: Bench-mark results on two high-dimensional locomotion tasks from OpenAI Gym environment *Hopper-v2* **(a)**, and *Half Cheetah-v2* **(b)**. The graphs depict the performance of the deep RL model on the two tasks as a function of time for different maximum trajectory lengths $T_{\max}$ within each episode. The plots are averaged over 5 random seeds. The policy performance is assessed every 5000 steps over 1000000 time steps.

5.7.2 The Impact of $T_{\max}$

We test the performance of attack classifiers against the target model for different values of $T_{\max}$ in a set of experiments. As the environment is unvarying, the value of $T_{\max}$ remains unchanged throughout each experiment. Our observations presented in Figure 5.3 show that as $T_{\max}$ increases, the performance of the attack classifiers in both individual and collective modes improves. Since MCC utilizes all four values in the confusion matrix, it provides a more reliable and robust measure compared to the other metrics (Table 5.2). Our results show consistent improvement of MCC across different values of $T_{\max}$ in both Hopper-v2 and Half Cheetah-v2 environments, which is consistent with the changes in ACC.

Maximum trajectory length $T_{\max}$ plays a significant role in the performance of deep RL models. Figure 5.4 illustrates the learning curves for the deep RL agent in Hopper-v2 (Figure 5.4(a)) and HalfCheetah-v2 (Figure 5.4(b)) for different values of $T_{\max}$. The deep RL policy is evaluated every 5000 time steps for the total number of 1000000 steps. The plots show that as $T_{\max}$ increases, the deep RL policy presents a consistently improved behaviour. As RL policy is the function that maps the visited states to the selected actions, a closer deep RL policy to the optimal policy corresponds to a more predictable relationship between the training and the output trajectories. We argue that this feature of deep RL policies contributes to the vulnerability of deep RL models that are trained on larger values of $T_{\max}$.

As the attack classifiers output membership probabilities, we determine the predicted binary label with respect to a range of acceptance thresholds $\theta = 0.1, 0.2, \ldots, 0.9$, and subsequently choose the threshold θ, at which the classifier shows the highest performance. Figure 5.5 depicts the sample ROC curves for HalfCheetah-v2 in individual (Figure 5.5(a)) and collective (Figure 5.5(b)) modes. The plots show that the larger $T_{\max}$ is, the attacker shows a better performance. The best result is obtained at $T_{\max} = 200$ in the collective mode (Figure 5.5(b)). We find that the acceptance threshold $\theta = 0.5$ yields the highest performance throughout all of our experiments.

5.7.3 Clipping Length Impact

In our experimental design, we further study the effect of varying the clipping length on the performance of the attack classifier. Clipping length determines the length of paired action trajectories: trajectories longer than the clipping length are trimmed, while shorter ones are extended via repeating the last action for the required number of times. Figure 5.6 illustrates a sample graph that shows the performance of the adversary in individual and collective modes in Hopper-v2 at $T_{\max} = 100$ for a range of clipping lengths. Our results show that the attacker's performance against the deep RL algorithm is relatively invariant with respect to the clipping length. This observation indicates that due to the temporal correlation between the transition

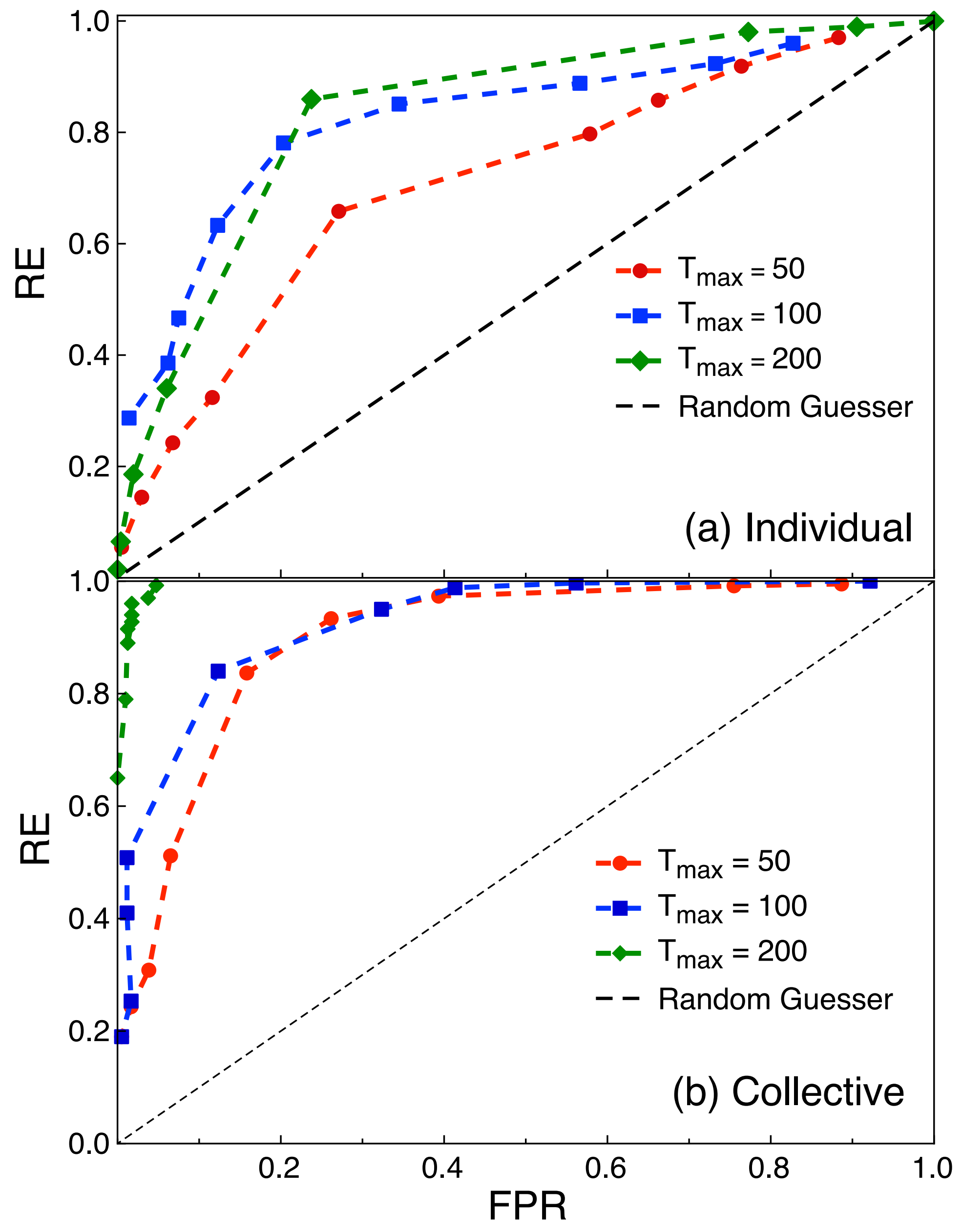


FIGURE 5.5: The receiver operator characteristic (ROC) curves of the membership inference attack in HalfCheetah-v2 in the individual **(a)** and collective **(b)** modes for different values of $T_{\max}$. We note that, the the trained attack classifiers are evaluated on the balanced test data and thus, we choose 50% as the baseline for random guess.

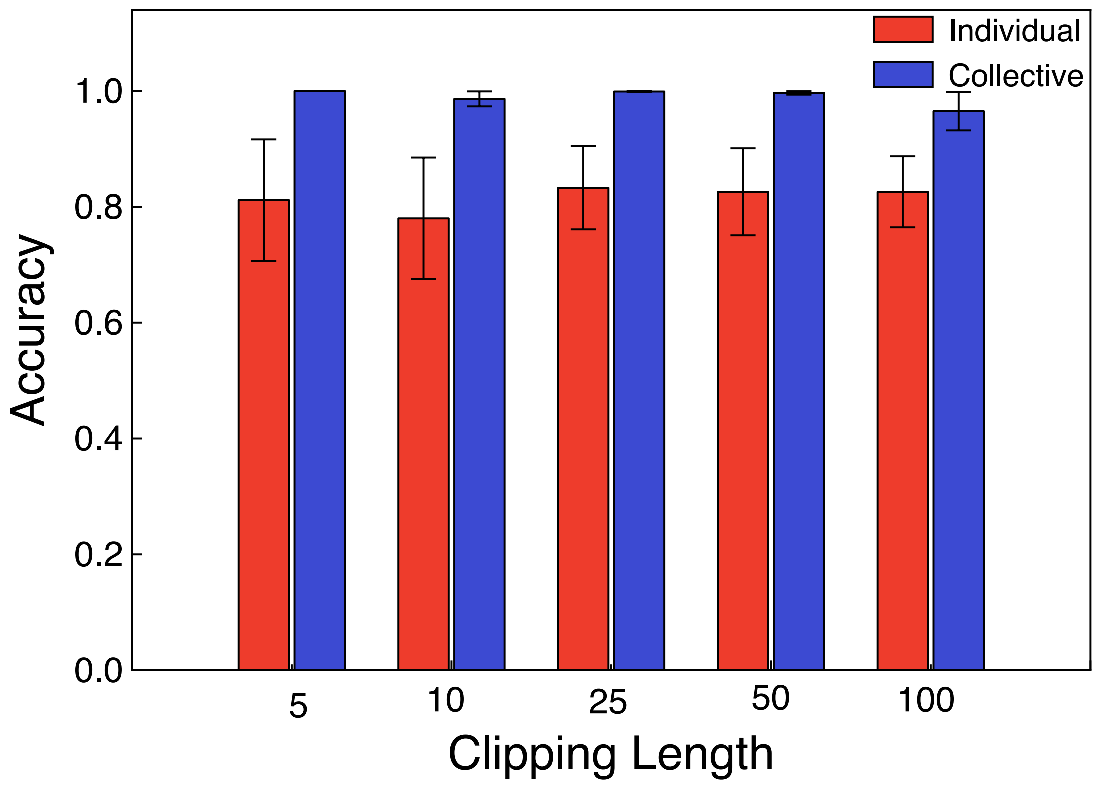


FIGURE 5.6: The accuracy of data inference in Hopper-v2 at $T_{\max} = 100$ for different clipping lengths in individual and collective attack modes. Each data point is determined from the average result of 5 separate runs. The error bars depict the error on the mean for accuracy measurements for the corresponding runs.

tuples in a trajectory, the first few tuples carry sufficient information for learning the relationship between the paired trajectories.

5.7.4 Temporal Correlation

The results presented so far exhibit the performance of the membership inference attacks against deep reinforcement learning as a result of training the attack classifiers on the temporally correlated data collected from the training set and output of the deep RL model. At this point, a question may arise: how do we know that the high performance of the MIAs is the consequence of temporal correlation in the data set?

To answer this question, we have performed a set of experiments, where before the data augmentation phase, the temporal correlation between the deep RL training trajectories is broken. In particular, we first decorrelate the trajectories through shuffling the tuples of trajectory used in training the deep RL model and subsequently store the decorrelated transition tuples in an auxiliary buffer. In the next step, we generate trajectories of the desired length by sampling

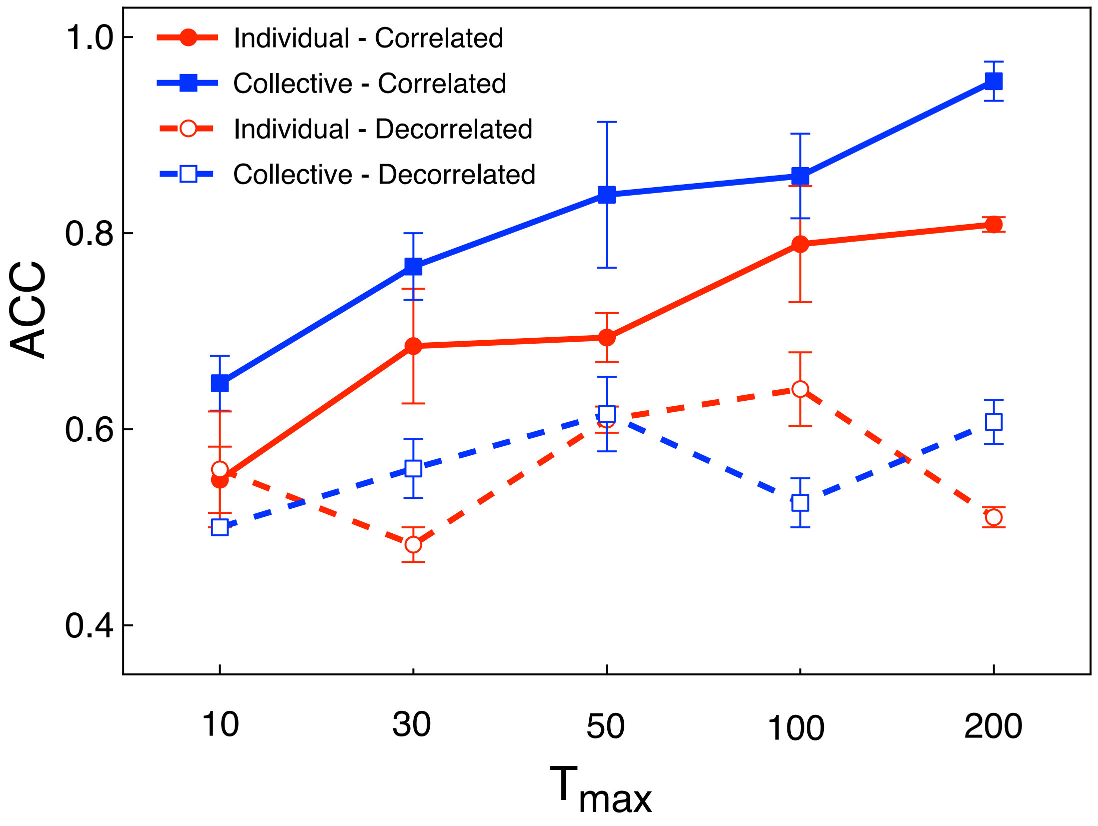


FIGURE 5.7: Comparison of the membership inference attack accuracy between correlated and decorrelated settings for HalfCheetah-v2.

actions uniformly from the buffer. Finally, we pass the collection of decorrelated trajectories to the data augmentation mechanism and train the attack classifiers with the paired trajectories in individual and collective modes. Figure 5.7 compares the accuracy of the membership inference attack in the correlated mode with that in the decorrelated mode. The plots depict that upon the decorrelation of the training trajectories, the adversary's accuracy in inferring RL training members decreases significantly. The results show that despite the inevitable input decorrelation imposed by the replay buffer mechanism in the training phase of off-policy deep RL models, the temporal correlation in the training trajectories is channeled to the model output data points. Thus, the attack classifiers trained on temporally correlated training data points exhibit higher accuracy in MIA than those trained on decorrelated trajectories.

5.8 Discussion

In this chapter, we designed and evaluated the first membership inference attack framework against off-policy deep reinforcement learning in collective and individual membership inference modes by exploiting the temporal correlations present in RL algorithms. We demonstrated the behavior of the proposed adversarial attack framework in complex high-dimensional locomotion tasks for different maximum trajectory lengths. The proposed framework reveals substantial vulnerability of a state-of-the-art off-policy deep RL model to black-box membership inference attacks. Moreover, we showed that reinforcement learning is significantly more vulnerable to membership inference attack in a collective setting than in the individual membership setting. In addition, the experimental results reveal that the maximum trajectory length, which is set by the environment, plays a significant role in the vulnerability of the deep reinforcement learning model to the membership inference attack. A longer maximum trajectories correspond to less privacy. Finally, our results reveal the role of temporal correlation in attack training. This type of information can be utilized very effectively to design high accuracy membership inference attacks against deep reinforcement learning. The results from this study highlight serious privacy concerns that may affect the widespread deployment of models resulting from deep reinforcement learning. It is important to investigate algorithmic solutions to this problem in in future work.

Chapter 6

Conclusions and Future Work

6.1 Main Findings

This dissertation developed algorithms and frameworks that address the challenges in designing private reinforcement learning algorithms. In Chapter 3, we introduced the first privacy-preserving policy evaluation algorithm under full MDP setting. Our theoretical analysis and empirical evaluation unveiled the trade-off between the privacy budget, regularization coefficient and the sample size. Under a particular selection schedule for the regularization coefficient with respect to the sample size, the upper bound we derived on the empirical excess risk explicitly demonstrated how the aforementioned parameters should be adjusted to minimize the empirical excess risk. Additionally, we developed the first application of the smooth sensitivity framework on the class of empirical risk minimization problems (ERMs) in which the loss function is non-Lipschitz. Moreover, we extended the natural model of neighbouring datasets from a supervised setting, where a data point is composed of a single regression target, to the reinforcement learning setting, where a data point is composed of a trajectory of transition tuples with multiple regression targets. Our experimental results show that in a non-tabular setting, a more aggressive function approximation provides a better rate of convergence in both of the proposed private algorithms. This observation is consistent with the results presented by some previous studies on privacy-preserving linear regression (Wang, 2018; Sheffet, 2019).

The sub-sampling technique proposed in Chapter 4 successfully improved the utility of privacy-preserving linear queries. From a theoretical perspective, we introduced new sensitivity and utility measures tailored to our proposed utility-amplification framework. We developed theorems that show how to adaptively apply the proposed sub-sampling subroutine in different privacy regimes, for different sample size choices and privacy parameters. In particular, we showed that if the data distribution satisfies specific properties with respect to the number of sub-samples and the sub-sample size, the utility of the proposed sub-sample and average mechanism improves compared with that of the classic Gaussian mechanism. In the generic setting with no assumption on the underlying data distribution, we explicitly derived the required relationship between the sample size, number of sub-samples, and the sub-sample size. Empirically, our proposed sub-sampling framework successfully improves the utility of the DP-LSL and DP-LSW algorithms introduced in Chapter 3 in a chain MDP baseline. We tested the performance of our utility amplification framework in comparison to the DP-LSL and DP-LSW algorithms in different privacy regimes. The results we presented showcase the adaptability of the proposed algorithmic approach to different privacy parameters.

Finally, in Chapter 5 we introduced the first membership inference attack framework against deep reinforcement learning algorithms. We studied some of the factors contributing to the vulnerability of off-policy deep reinforcement learning, including the learning state of the target policy with respect to the optimal policy, individual versus collective data point inference, and the temporal correlation in the attack classifier training set. Our results show that off-policy deep reinforcement learning is significantly vulnerable to black-box membership inference attacks. In particular, we demonstrate that as the maximum trajectory length in each episode increases and the target policy gets closer to the optimal policy, a more meaningful (less noisy) relationship can be built between the input training data point and the output trajectories. Thus, the reinforcement learning model shows more vulnerability to membership inference attacks as this trajectory length increases, as the attacker can better exploit the temporal correlations inherent in trajectories generated in RL, in order to provide accurate identification. Moreover, our results show that deep reinforcement learning exhibits a higher vulnerability to membership inference attacks in collective mode, where the adversary aims at inferring the membership of a collection of data points, as opposed to the individual mode. Because the temporal correlation within the training trajectories of the attack classifier plays an important role in the adversary's learning of the relationship between the paired trajectories, breaking the temporal correlation in one of the paired trajectories leads to a lower membership inference accuracy. This observation could be used as a starting point for the design of deep RL algorithms that are less vulnerable to membership inference attacks.

6.2 Future directions

Our work presents multiple directions for future studies, both in designing privacy-preserving reinforcement learning algorithms and testing the vulnerability of reinforcement learning algorithms to membership inference attacks.

6.2.1 Algorithm Design Extensions

One of the most important families of reinforcement learning algorithms is the temporal difference (TD) learning algorithms. Adopting or introducing the proper privacy definition tailored to the inherent assumptions of TD algorithms is an important research direction that would extend the results of this thesis, as discussed at the end of Chapter 3.

Our experiments on private Monte Carlo policy evaluation with linear function approximation were very encouraging, and during this work, we noticed that one approach which could be beneficial both from the point of view of privacy as well as from the direction of achieving good utility is the use of random projections. In particular, the use of the Johnson–Lindenstrauss transform has already been studied in designing adaptive differentially private linear regression algorithms (Wang, 2018) and differentially private ordinary least square methods (Sheffet, 2019), and our approach could be extended in this direction. Specifically, our notion of neighbourhood would still hold, and some of our theorems can likely be combined with the results mentioned above without too much trouble. Studies on non-private least-square temporal difference learning algorithms (Ghavamzadeh et al., 2010; Li et al., 2018) suggests that random projections can be useful for TD algorithms as well, so we could explore this direction in the design of differentially private least-squares TD learning algorithms.

Our final goal is to design differentially private algorithms for the whole reinforcement learning problem. Policy evaluation is an important sub-problem, as it is typically used as a building block towards control algorithms (eg by leveraging policy improvement or policy gradients. We anticipate that by invoking the compositional properties that hold in many differential privacy approaches, the ideas and algorithms we presented could be plugged into any reinforcement learning algorithm. For example, one could consider using our algorithms to derive a differentially private version of Q-learning by estimating a state-action value function instead of a state value function. Our approach could guarantee the privacy of the policy evaluation part, but careful further thinking would be required to understand how the fact that the agent is actively choosing actions may provide further privacy leaks to be mitigated. Some ideas from the literature on differential privacy for bandits could be used as a starting point, but would require significant extensions to account for correlations between states and actions inside trajectories. We hope to investigate this and other alternatives to achieve differentially private reinforcement learning algorithms in future work.

6.2.2 Membership Inference Attack Extensions

The work in Chapter 5 is just a first step in developing our understanding of membership inference attacks for RL. It focuses less on theory than the rest of the thesis (and the rest of the differential privacy field) and more on working with RL algorithms that are most relevant for practice. Improving the theoretical understanding of how attacks could be both designed and counteracted, by looking at easier cases (such as tabular and linear function approximation) could be quite useful.

One of the directions in which our study on membership inference attacks can be extended is training the attack classifier in a white-box setting, where the adversary benefits from knowing the internal structure of the deep reinforcement learning model. For instance, the change in the gradient norm of the network parameters captures the sensitivity of the target policy to change in the private training set. The adversary could use this auxiliary information to design attack classifiers. Moreover, system designers can benefit from our proposed attack framework to measure the privacy level of algorithms and determine the required level of privacy (*i.e.* privacy parameters) in the industrial deployment of privacy-preserving reinforcement learning algorithms.

Ultimately, we believe that privacy considerations will be just as important to the deployment of RL algorithms in domains such as health care, finance or education, as the need to have robust generalization or efficient exploration. The theoretical properties of RL algorithms are quite a bit more difficult to establish than for supervised learning as well, so this research area also raises interesting mathematical puzzles. While in this thesis we took some early steps in the direction of building private RL algorithms, we hope our work will inspire more researchers to get into this field.

Bibliography

Borja Balle, Maziar Gomrokchi, and Doina Precup. Differentially private policy evaluation. In *International Conference on Machine Learning*, pages 2130–2138, 2016.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

Susan Amin, Maziar Gomrokchi, Hossein Aboutalebi, Harsh Satija, and Doina Precup. Locally persistent exploration in continuous control tasks with sparse rewards. *arXiv preprint arXiv:2012.13658*, 2020.

Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Scott Mayer McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg C Corrado, Ara Darzi, et al. International evaluation of an ai system for breast cancer screening. *Nature*, 577(7788):89–94, 2020.

Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

Marie Douriez, Harish Doraiswamy, Juliana Freire, and Cláudio T Silva. Anonymizing nyc taxi data: Does it matter? In *2016 IEEE international conference on data science and advanced analytics (DSAA)*, pages 140–148. IEEE, 2016.

Vijay Pandurangan. On taxis and rainbows: Lessons from nyc's improperly anonymized taxi logs. *Medium. Accessed November*, 30:2015, 2014.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

Luc Rocher, Julien M Hendrickx, and Yves-Alexandre De Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature communications*, 10 (1):1–9, 2019.

Graham Cormode. Personal privacy vs population privacy: learning to attack anonymization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1253–1261, 2011.

Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 17–32, 2014.

Ninghui Li, Wahbeh Qardaji, Dong Su, Yi Wu, and Weining Yang. Membership privacy: a unifying framework for privacy definitions. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 889–900, 2013.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

Justin Brickell and Vitaly Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 70–78, 2008.

Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-35907-9, 978-3-540-35907-4. doi: 10.1007/11787006_1. URL `http://dx.doi.org/10.1007/11787006_1`.

Arik Friedman and Assaf Schuster. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–502, 2010.

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

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876, pages 265–284. Springer, 2006.

Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

Zhanglong Ji, Zachary C Lipton, and Charles Elkan. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*, 2014.

Pathum Chamikara Mahawaga Arachchige, Peter Bertok, Ibrahim Khalil, Dongxi Liu, Seyit Camtepe, and Mohammed Atiquzzaman. Local differential privacy for deep learning. *IEEE Internet of Things Journal*, 7(7):5827–5842, 2019.

Noman Mohammed, Dima Alhadidi, Benjamin CM Fung, and Mourad Debbabi. Secure two-party differentially private data release for vertically partitioned data. *IEEE transactions on dependable and secure computing*, 11(1):59–71, 2013.

Dima Alhadidi, Noman Mohammed, Benjamin Fung, and Mourad Debbabi. Secure distributed framework for achieving ε-differential privacy. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 120–139. Springer, 2012.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Herve Jegou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, pages 5558–5567, 2019.

Hongsheng Hu, Zoran Salcic, Gillian Dobbie, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *arXiv preprint arXiv:2103.07853*, 2021.

Du Su, Hieu Tri Huynh, Ziao Chen, Yi Lu, and Wenmiao Lu. Re-identification attack to privacy-preserving data analysis with noisy sample-mean. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1045–1053, 2020.

S Kevin Zhou, Hoang Ngan Le, Khoa Luu, Hien V Nguyen, and Nicholas Ayache. Deep reinforcement learning in medical imaging: A literature review. *arXiv preprint arXiv:2103.05115*, 2021.

Iwao Maeda, David deGraw, Michiharu Kitano, Hiroyasu Matsushima, Hiroki Sakaji, Kiyoshi Izumi, and Atsuo Kato. Deep reinforcement learning in agent based financial market simulation. *Journal of Risk and Financial Management*, 13(4):71, 2020.

Baoxiang Wang and Nidhi Hegde. Privacy-preserving q-learning with functional noise in continuous spaces. In *Advances in Neural Information Processing Systems*, pages 11323–11333, 2019.

Vinith M Suriyakumar, Nicolas Papernot, Anna Goldenberg, and Marzyeh Ghassemi. Chasing your long tails: Differentially private prediction in health care settings. *arXiv preprint arXiv:2010.06667*, 2020.

Giuseppe Vietri, Borja Balle, Akshay Krishnamurthy, and Steven Wu. Private reinforcement learning with pac and regret guarantees. In *ICML 2020*, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

Csaba Szepesvári. *Algorithms for reinforcement learning*. Morgan & Claypool Publishers, 2010.

Richard Bellman. Dynamic programming. 1957.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Shangtong Zhang and Richard S Sutton. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*, 2017.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Ruishan Liu and James Zou. The effects of memory replay in reinforcement learning. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 478–485. IEEE, 2018.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pages 3061–3071. PMLR, 2020.

Yonghui Xiao, Li Xiong, and Chun Yuan. Differentially private data release through multidimensional partitioning. In *Workshop on Secure Data Management*, pages 150–168. Springer, 2010.

Noman Mohammed, Rui Chen, Benjamin CM Fung, and Philip S Yu. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–501, 2011.

Rui Chen, Noman Mohammed, Benjamin CM Fung, Bipin C Desai, and Li Xiong. Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment*, 4(11):1087–1098, 2011.

Rui Chen, Benjamin Fung, Philip S Yu, and Bipin C Desai. Correlated network data publication via differential privacy. *The VLDB Journal*, 23(4):653–676, 2014.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 94–103. IEEE, 2007.

Nikita Mishra and Abhradeep Thakurta. (nearly) optimal differentially private stochastic multi-arm bandits. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 592–601. AUAI Press, 2015.

Aristide CY Tossou and Christos Dimitrakakis. Algorithms for differentially private multi-armed bandits. In *AAAI*, pages 2087–2093, 2016.

Roshan Shariff and Or Sheffet. Differentially private contextual linear bandits. In *NeurIPS*, 2018.

Debabrota Basu, Christos Dimitrakakis, and Aristide Tossou. Differential privacy for multi-armed bandits: What is it and what is its cost? *arXiv preprint arXiv:1905.12298*, 2019.

Abhimanyu Dubey. No-regret algorithms for private gaussian process bandit optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2062–2070. PMLR, 2021.

Rob Hall, Alessandro Rinaldo, and Larry Wasserman. Differential privacy for functions and functional data. *The Journal of Machine Learning Research*, 14(1):703–727, 2013.

Jennifer Kulynych and Henry T Greely. Clinical genomics, big data, and electronic medical records: reconciling patient rights with research when privacy and science collide. *Journal of Law and the Biosciences*, 4(1):94–132, 2017.

Deborah Stiles and Paul S Appelbaum. Cases in precision medicine: concerns about privacy and discrimination after genomic sequencing. *Annals of internal medicine*, 170(10):717–721, 2019.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 117–126. ACM, 2015.

Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *NCSS*, 2019.

Yunhui Long, Vincent Bindschaedler, and Carl A Gunter. Towards measuring membership privacy. *arXiv preprint arXiv:1712.09136*, 2017.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1):133–152, 2019a.

Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.

Xinlei Pan, Weiyao Wang, Xiaoshuai Zhang, Bo Li, Jinfeng Yi, and Dawn Song. How you act tells a lot: Privacy-leaking attack on deep reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 368–376, 2019.

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

Davide Chicco, Niklas Tötsch, and Giuseppe Jurman. The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData mining*, 14(1):1–22, 2021.

Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, pages 289–296, 2009.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *Journal of Privacy and Confidentiality*, 4(1):4, 2012.

Prateek Jain and Abhradeep Thakurta. Differentially private learning with kernels. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 118–126, 2013.

Abhradeep Guha Thakurta and Adam Smith. Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Conference on Learning Theory*, pages 819–850, 2013.

Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2007.

Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis, 2011. URL `http://www.cse.psu.edu/~ads22/pubs/NRS07/`.

Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000.

Prateek Jain and Abhradeep Guha Thakurta. (near) dimension independent risk bounds for differentially private learning. In *Proceedings of The 31st International Conference on Machine Learning*, pages 476–484, 2014.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

Justin A Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2):233–246, 2002.

Mohammad Ghavamzadeh, Alessandro Lazaric, Odalric Maillard, and Rémi Munos. Lstd with random projections. In *Advances in Neural Information Processing Systems*, pages 721–729, 2010.

Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: tight analyses via couplings and divergences. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6280–6290, 2018.

Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1226–1235. PMLR, 2019.

Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 521–532. IEEE, 2018.

Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.

Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *Annual International Cryptology Conference*, pages 638–667. Springer, 2019.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

J Ross Quinlan et al. Bagging, boosting, and c4. 5. In *Aaai/iaai, Vol. 1*, pages 725–730, 1996.

James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Differentially private bagging: Improved utility and cheaper privacy than subsample-and-aggregate. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Yu-Xiang Wang. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. *arXiv preprint arXiv:1803.02596*, 2018.

Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.

Jonathan Ullman. Cs7880: Rigorous approaches to data privacy, spring 2017, assignment 1. 2017. URL `http://www.ccs.neu.edu/home/jullman/cs7880s17/HW1sol.pdf`.

Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 51–60. IEEE, 2010.

Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4:61–84, 2017.

Tensorflow privacy library. `https://github.com/tensorflow/privacy`. Accessed: 2021-09-08.

Pytorch opacus library. `https://opacus.ai/`. Accessed: 2021-09-08.

Thomas Humphries, Matthew Rafuse, Lindsey Tulloch, Simon Oya, Ian Goldberg, Urs Hengartner, and Florian Kerschbaum. Differentially private learning does not bound membership inference. *arXiv preprint arXiv:2010.12112*, 2020.

Zuobin Ying, Yun Zhang, and Ximeng Liu. Privacy-preserving in defending against membership inference attacks. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, pages 61–63, 2020.

Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *arXiv preprint arXiv:2007.07646*, 2020.

Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International Conference on Machine Learning*, pages 1964–1974. PMLR, 2021.

Borislav Mavrin, Hengshuai Yao, and Linglong Kong. Deep reinforcement learning with decorrelation. *arXiv preprint arXiv:1903.07765*, 2019.

Yunhui Long, Lei Wang, Diyue Bu, Vincent Bindschaedler, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen. A pragmatic approach to membership inferences on machine learning models. In *2020 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 521–534, 2020. doi: 10.1109/EuroSP48549.2020.00040.

Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1):133–152, 2019b. doi: doi:10.2478/popets-2019-0008. URL `https://doi.org/10.2478/popets-2019-0008`.

Benjamin Hilprecht, Martin Härterich, and Daniel Bernau. Monte carlo and reconstruction membership inference attacks against generative models. *Proc. Priv. Enhancing Technol.*, 2019(4):232–249, 2019.

Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. Gan-leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 343–362, 2020.

Congzheng Song and Vitaly Shmatikov. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–206, 2019.

Sorami Hisamoto, Matt Post, and Kevin Duh. Membership inference attacks on sequence-to-sequence models: Is my data in your machine translation system? *Transactions of the Association for Computational Linguistics*, 8:49–63, 2020.

Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. What does the crowd say about you? evaluating aggregation-based location privacy. *Proceedings on Privacy Enhancing Technologies*, 4:76–96, 2017.

Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Measuring membership privacy on aggregate location time-series. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2):1–28, 2020.

Richard S Sutton. Temporal credit assignment in reinforcement learning. 1985.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596, 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

Haiguang Wen, Junxing Shi, Wei Chen, and Zhongming Liu. Deep residual network predicts cortical representation and organization of visual features for rapid categorization. *Scientific reports*, 8(1):1–17, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11): 3212–3232, 2019.

Or Sheffet. Old techniques in differentially private linear regression. In *Algorithmic Learning Theory*, pages 789–827, 2019.

Haifang Li, Yingce Xia, and Wensheng Zhang. Finite sample analysis of lstd with random projections and eligibility traces. *arXiv preprint arXiv:1805.10005*, 2018.