Adaptive and Efficient Optimization Algorithms for Differentially Private Machine Learning

by

Chen Chen

(Under the Direction of Jaewoo Lee)

Abstract

In this dissertation, the problem of solving an empirical risk minimization (ERM) task is tackled in a differentially private (DP) way. The ultimate goal is to propose DP algorithms with high utility and efficiency under high privacy range. Several algorithms are proposed, each solving the problem through different methods under different assumptions. The first algorithm applies on strongly convex and smooth objectives. It utilizes a sensitivity calculation strategy to scale the sensitivity of the model after multiple iterations of randomly shuffled mini-batch permutation stochastic gradient descent through contraction mapping. Periodic averaging technique is applied to fasten convergence and further reduce the sensitivity to decrease the scale of noise perturbation. The second algorithm applies on convex objectives with non-smooth regularization. It utilizes a stochastic version of ADMM algorithm to split the smooth and non-smooth portions of the problem, and both sub-problems have closed-form solutions. Two versions are proposed based on this technique: One version is based on gradient perturbation, and applies the privacy amplification result to reduce the noise. The other version is based on output perturbation and sensitivity calculation. The third algorithm applies on both convex and non-convex problems. It spends a portion of the privacy budget on objective evaluations to perform adaptive step size selection through Armijo line search, and the privacy is bounded by a non-trivial utilization of the sparse vector technique. It also adjusts the per-iteration privacy budget during runtime, according to the reliability of the noisy gradient. The last algorithm is an application of differential privacy on the information retrieval task of a ranking model. By observing that only the preference labels come from the user, document vectors are considered as public information, which can be used to scale the sensitivity of the accumulated gradients.

Extensive experimental results show the effectiveness of all the proposed algorithms, comparing to private baselines. This dissertation advances the state-of-the-art in the area of DP-ERM, and provide new insights of combining ERM technique and privacy protection techniques to train a machine learning model privately, which greatly benefits the machine learning practitioners to release their model with strong privacy guarantee.

INDEX WORDS: [Differential Privacy, Empirical Risk Minimization, Convex Optimization, Neural Network, Sensitivity Calculation, Non-smooth Regularization, Step Size Selection, Adaptive Budget Allocation, Ranking]

Adaptive and Efficient Optimization Algorithms for Differentially Private Machine Learning

by

Chen Chen

B.M., Peking University, China, 2011

M.S., University of Georgia, 2015

M.S., University of Georgia, 2018

A Dissertation Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2020

Adaptive and Efficient Optimization Algorithms for Differentially Private Machine Learning

by

Chen Chen

Major Professor: Jaewoo Lee

Committee: Roberto Perdisci
Khaled Rasheed

Electronic Version Approved:

Ron Walcott
Dean of the Graduate School
The University of Georgia
December 2020

Dedication

Dedicate to my family, especially to my wife, Yan.

Acknowledgments

I want to give my sincere appreciation to my advisor, Dr. Jaewoo Lee, for his long-time support of my research and academic growth, including directing me to necessary background knowledge, searching for innovative research ideas and ways of improving, guidance in mathematics and coding, suggestions in my writing and presenting skills, etc. His continuous encouragement make me overcome all the difficulties during my PhD study. I also acknowledge my committee members, Dr. Roberto Perdisci and Dr. Khaled Rasheed, in providing me guidance for both my research and my teaching and presentation. Thanks to all the faculty members in the department of computer science in academic advising and course instructions. Last but not least, thanks to my family members for all the loving memories during my graduate study.

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction and Summary

With the omnipresence of intelligent agents in modern society, privacy protection becomes an essential factor for the welfare of human beings. It not only raises legal issues of data and model release, in domains where sensitive information are involved, such as education, finance, and healthcare, but also affects people's confidence on the trustworthiness of modern technology. Most AI techniques which raise privacy issues are based on machine learning models trained on data collected from humans, but utilizing those dataset is essential for the success of personalized AI. Therefore, it is necessary to protect private information while releasing an AI product. Differential privacy [1] is the state-of-the-art technique of privacy preserving machine learning/data releasing, which provides protections against strong adversaries, thanks to its strict mathematical definition.

The topic of my dissertation is about differentially private empirical risk minimization (ERM) algorithms. It draws considerations into calculating the sensitivity of strongly convex and smooth objectives, tackling non-smooth regularization by taking advantage of ADMM algorithm, and splitting part of privacy budget for adaptively selecting step sizes. Each chapter of my dissertation seeks the privacy protection on one specific type of ERM problems, and utilizes different techniques to guarantee the privacy, but all the proposed algorithms fall into the recent Rényi differential privacy framework.

The second chapter contains the definitions of differential privacy and Rényi differential privacy. In the third chapter, a Rényi Differentially Private stochastic gradient descent (SGD) algorithm for convex ERM is presented. The algorithm uses output perturbation technique and leverages intrinsic randomness of SGD, which creates a "randomized sensitivity", in order to reduce the necessary amount of noise to be added. One advantage of output perturbation is its ability to incorporate a periodic averaging step which

serves to further reduce the sensitivity. Moreover, as a technique to reduce the well-known oscillating behavior of SGD near the optimum, periodic averaging with warm re-start also improves performance. The proposed algorithm, RSGD-AR, leverages the averaging step for these two sides of benefits. Rényi Differential Privacy (RDP) can be used to provide (ϵ, δ)-differential privacy guarantees and hence provide a comparison with prior work. An empirical evaluation demonstrates that the proposed method outperforms prior methods on differentially private ERM.

One limit of the RSGD-AR algorithm is that it can only be applied to smooth objective functions. In the fourth chapter, the problem of optimizing composite objective functions consisting a convex differentiable loss function and a non-smooth regularization term, such as L_1 norm or nuclear norm, is considered under RDP. To solve the problem, two stochastic ADMM algorithms are proposed, one based on gradient perturbation and the other based on output perturbation. Both algorithms decompose the original problem into sub-problems which have closed-form solutions. The first algorithm, SSADMM, applies the recent privacy amplification result for RDP to reduce the amount of noise added. The second algorithm, MPADMM, numerically computes the sensitivity of ADMM variable updates and releases the updated parameter vector at the end of each epoch. The performance of the two algorithms was compared with baseline algorithms on both real and simulated datasets. Experimental results show that, in high privacy regimes (small ϵ), SSADMM and MPADMM outperform baseline algorithms in terms of classification and feature selection performance, respectively.

In the fifth chapter, a private algorithm with adaptive privacy budget allocation is proposed, which can be applied on a broader range of optimization problems. The performance of private gradient-based optimization algorithms is highly dependent on the choice of step size (or learning rate) which often requires non-trivial amount of tuning. A stochastic variant of classic backtracking line search algorithm is proposed in this chapter that satisfies Rényi differential privacy. Specifically, the proposed algorithm adaptively chooses the step size satisfying the Armijo condition (with high probability) using noisy gradients and function estimates. Furthermore, to improve the probability with which the chosen step size satisfies the condition, it adjusts the per-iteration privacy budget during run-time, according to the reliability of noisy gradient. A naive implementation of the backtracking search algorithm may end up using unacceptably large privacy budget as the ability of adaptive step size selection comes at the cost of extra function evaluations. The proposed algorithm avoids this problem by using the sparse vector technique combined with the recent privacy amplification lemma. Furthermore, a privacy budget adaptation strategy is in-

troduced in which the algorithm adaptively increases the budget when it detects that directions pointed by consecutive gradients are drastically different. Extensive experiments on both convex and non-convex problems show that the adaptively chosen step sizes allow the proposed algorithm to efficiently use the privacy budget and show competitive performance against existing private optimizers.

In the last chapter, an application of differentially private ERM was proposed on the learning to rank model. Consider the supervised learning to rank task as a general optimization problem, existing private optimization techniques can be applied on ranking models. However, most datasets used to train a ranking model are session based and have their own properties. Root on the characteristics of an information retrieval dataset, a differentially private framework was proposed to protect the information of users' preference on the candidate documents. Since for many ranking applications, only the ground truth scores representing preference come from the users, this framework considers the document feature vectors as public information, which can be used to measure the sensitivity of the users' information. One property of the BayesRank algorithm, that the users' evaluations are only accessed through the NDCG calculation, is utilized in our privacy analysis, and a sensitivity calculation technique is presented, which can be used to calibrate the noise injected at each iteration of training. Spectral normalization technique was further integrated into this private algorithm on neural network models to regularize the model parameters. Experimental results are performed comparing different approaches of learning to rank models with different privacy, and the sensitivity calculation technique can improve the utility of the private model.

Chapter 2

Preliminaries on Differential Privacy

Two datasets D and D' are called *neighboring*, denoted as $D \sim D'$, if D and D' are differed by one record[1].

2.1 Differential Privacy

Differential privacy is so far the golden-standard technique used in protecting the privacy of a sensitive dataset. It is formally defined as follows:

Definition 2.1 ((ϵ, δ)-Differential Privacy (DP)). *[2] [1] Given privacy parameters $\epsilon \geq 0, 0 \leq \delta \leq 1$, a randomized mechanism (algorithm) $\mathcal{M}$ satisfies (ϵ, δ)-differential privacy if for every event $S \subseteq range(\mathcal{M})$, and for all pairs of neighboring datasets $D \sim D'$,*

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

The case where $\delta = 0$ is referred to as *pure* differential privacy and the case where $\delta > 0$ is referred to as *approximate* differential privacy.

Intuitively, privacy is guaranteed by limiting the impact of every individual element to the output of the mechanism. Pure differential privacy limits the ability of an attacker to make inferences about the

[1] There are two typical definitions of neighboring datasets: replacement definition and addition/subtraction definition. The exact definition used for each algorithm proposed in this dissertation is specified in the "preliminaries" section of each chapter.

specific record of any individual. Approximate differential privacy provides the same guarantee with high chance, and allows failure of privacy protecting (e.g. release of the raw data) to occur with probability at most δ. This event is known informally as the "all-bets-are-off" scenario. In practice, δ needs to be set as criptographically small.

While approximate DP is a relaxation of pure DP, some other relaxations also exists, such as zero-concentrated differential privacy (zCDP) [3] and Rényi Differential Privacy (RDP) [4]. These relaxations do not have such semantic meanings as approximate DP, but they are shown to stand between pure and approximate DP: they provide weaker protection than pure DP, but stronger protection than approximated DP, for any given $\delta > 0$.

2.2 Rényi Differential Privacy

Define $Z = \frac{\Pr[\mathcal{M}(D) \in S]}{\Pr[\mathcal{M}(D') \in S]}$ as the privacy loss random variable. While pure differential privacy requires that the constraint $e^{-\epsilon} \leq Z \leq e^{\epsilon}$ always holds, Rényi differential privacy (RDP) constrains it by the Rényi divergence:

Definition 2.2 (Rényi Divergence)**.** *Let P_1 and P_2 be probability distributions over a set Ω and let $\alpha \in (1, \infty)$. Rényi α-divergence $\mathbb{D}_\alpha$ is defined as:*

$$\mathbb{D}_\alpha[P_1 \| P_2] := \frac{1}{\alpha - 1} \log(\mathbb{E}_{x \sim P_2}[P_1(x)^\alpha P_2(x)^{-\alpha}]) \,. \tag{2.2}$$

Rényi differential privacy requires two parameters: a moment (order) α and a parameter ϵ that bounds the moment.

Definition 2.3 ((α, ϵ)-RDP)**.** *[4] Given a privacy parameter $\epsilon \geq 0$, and an $\alpha \in (1, +\infty)$, a randomized mechanism (algorithm) $\mathcal{M}$ satisfies (α, ϵ)-Rényi Differential Privacy (RDP) if for neighboring datasets D and D',*

$$\mathbb{D}_\alpha[\mathcal{M}(D) \| \mathcal{M}(D')] \leq \epsilon \,. \tag{2.3}$$

Note that when $\alpha = +\infty$, Rényi divergence becomes max divergence and (α, ϵ)-RDP coincides with pure $(\epsilon, 0)$-DP.

One method to achieve RDP is through the Gaussian mechanism: when a query $q(D)$ is taken over the dataset, the Gaussian mechanism adds an appropriately scaled Gaussian noise $\gamma \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_p)$, and release perturbed $q(D) + \gamma$.

Lemma 2.1 (Gaussian mechanism)**.** *[4] Let $q : \mathcal{D} \to \mathbb{R}^k$ be a vector-valued function over datasets. Let $\mathcal{M}$be a mechanism that releases $\mathcal{N}(q(D), \sigma^2 \mathbb{I}_k)$. Then for any pair of neighboring datasets D and D' and any $\alpha \in (1, +\infty)$:*

$$\mathbb{D}_\alpha[\mathcal{M}(D) \| \mathcal{M}(D')] \leq \alpha \Delta_2^2(q)/(2\sigma^2)\,.$$

In particular, if $\sigma^2 = \alpha \Delta_2^2(q)/(2\epsilon)$, then $\mathcal{M}$ satisfies (α, ϵ)-RDP.

Therefore, when scale the variance $\sigma^2 = \alpha \Delta_2^2(q)/(2\epsilon)$, then $\mathcal{M}$ satisfies (α, ϵ)-RDP. Gaussian mechanism relies on the L_2 sensitivity $\Delta_2^2(q)$.

Definition 2.4 (L_1 (resp. L_2) sensitivity)**.** *Let $q : \mathcal{D}^n \to \mathbb{R}^k$ be a vector-valued function over datasets. The L_1 (resp. L_2) sensitivity of q, denoted as $\Delta_1(q)$ ($\Delta_2(q)$), is defined as*

$$\Delta_r(q) = \sup_{D \sim D'} \|q(D) - q(D')\|_r\,,$$

for $r = 1$ (resp. $r = 2$), where the sup *is over all possible pairs of neighboring datasets.*

Many mechanisms, like the Gaussian mechanism, make the algorithm $\mathcal{M}$ satisfy (α, ϵ)-RDP for a series of α, so we can use $\epsilon(\alpha)$ to denote the privacy ϵ under moment α.

While the semantics of RDP are still an area of research, it is currently used in practice through conversion to approximate DP [4].

Proposition 2.1 (Conversion from RDP to (ϵ, δ)-DP)**.** *[4] If $\mathcal{M}$ satisfies $(\alpha, \epsilon(\alpha))$-RDP, then it satisfies $(\epsilon(\delta), \delta)$-DP for $\epsilon(\delta) \geq \epsilon(\alpha) + \frac{\log(1/\delta)}{\alpha - 1}$.*

Therefore, to compare an RDP algorithm with an (ϵ, δ)-DP algorithm, one can do the following. First, one chooses an ϵ' and a very small $\delta' > 0$ with the intention of providing strictly stronger privacy protections than (ϵ', δ')-DP. Then one uses the above conversion result to find corresponding α and $\epsilon(\alpha)$ parameters for $(\alpha, \epsilon(\alpha))$-RDP.

Similar as DP, RDP has below composition properties:

Lemma 2.2 (RDP composition)**.** *[4] For randomized mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ applied on dataset D, if $\mathcal{M}_1$ satisfies (α, ϵ_1)-RDP and M_2 satisfies (α, ϵ_2)-RDP, then their composition $\mathcal{M}_1 \circ \mathcal{M}_2$ satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

The composition lemma can be used to quantify the privacy leak for a mechanism with multiple data accesses.

In many especially empirical risk minimization algorithms, it is common that the mechanism is taken over a randomized subsample of the dataset B, instead of the whole dataset D. Then, applying a private mechanism on the subsample B would satisfy $(\alpha, \epsilon(\alpha))$-RDP with respect to B. Due to the subsampling procedure, the mechanism would satisfy an amplified privacy with respect to the whole dataset D, as given by the following lemma.

Lemma 2.3 (RDP for subsampled mechanism: sampling without replacement)**.** *[5] For a randomized mechanism $\mathcal{M}$ and a dataset $D \sim \mathcal{D}^n$, if $\mathcal{M}$ satisfies $(\alpha, \epsilon(\alpha))$-RDP with respect to B, where B is a subsample of D by subsampling without replacement m datapoints from D (denote $q = m/n$ as sampling ratio). Then $\mathcal{M}$ satisfies $(\alpha, \epsilon'(\alpha))$-RDP with respect to D for any integer $\alpha \geq 2$, where*

$$\epsilon'(\alpha) \leq \frac{1}{\alpha - 1} \log \left(1 + q^2 \binom{\alpha}{2} \min \left\{4(e^{\epsilon(2)} - 1), 2e^{\epsilon(2)}\right\} + \sum_{j=3}^{\alpha} q^j \binom{\alpha}{j} 2e^{(j-1)\epsilon(j)}\right).$$

When the dataset D is privately accessed through a Poisson subsampling procedure, then the overall privacy can be amplified the the following lemma.

Lemma 2.4 (RDP for subsampled mechanism: Poisson sampling)**.** *[6] For a randomized mechanism $\mathcal{M}$ and a dataset $D \sim \mathcal{D}^n$, if $\mathcal{M}$ satisfies $(\alpha, \epsilon(\alpha))$-RDP with respect to B, where B is a subsample of D sampled by $B = \{\mathbf{d}_i | \iota_i = 1, \iota_i \overset{i.i.d}{\sim} Bernoulli(q)$ for $i \in [n]\}$. Then then $\mathcal{M}$ satisfies $(\alpha, \epsilon'(\alpha))$-RDP with respect to D for any integer $\alpha \geq 2$, where*

$$\begin{aligned}\epsilon'(\alpha) \leq \frac{1}{\alpha - 1} \log \Big\{&(1-q)^{\alpha-1}(\alpha q - q + 1) \\ &+ \binom{\alpha}{2} q^2 (1-q)^{\alpha-2} e^{\epsilon(2)} + 3 \sum_{l=3}^{\alpha} \binom{\alpha}{l} q^l (1-q)^{\alpha-l} e^{(l-1)\epsilon(l)}\Big\}.\end{aligned}$$

Chapter 3

Rényi Differentially Private ERM for Smooth Objectives

3.1 Motivation

There are three major ways to achieve differential privacy for empirical risk minimization algorithms: gradient perturbation, objective perturbation, and output perturbation. The latter two do not take the effort to composite privacy leak through multiple data accesses, but rely on careful analyses on the property of the objective function and/or the training mechanism. The first algorithms of each type were proposed by Chaudhuri et al. in one paper [7], which claimed that objective perturbation is more efficient than output perturbation. However, their output perturbation algorithm is based on an upper bound of the difference (sensitivity) of the optimal solutions when the convex objective function is applied on neighboring datasets, without looking into any specific optimization algorithm. This algorithm not only relies on a loose bound, but also suffers the same drawback as the objective perturbation mechanism does: privacy is achieved only if the problem is exactly solved, which is often impossible for complicated functions. Later, several extensions have been proposed on the output perturbation algorithms, such as the private full gradient descent algorithm from [8] and the private disjoint mini-batch permuting algorithm from [9], and both algorithms have significant heuristic improvements thanks to their tighter sensitivity calculation. This chapter is a further extension on this route, where an innovative "randomized

sensitivity" is calculated and used to achieve privacy through the analysis of Rényi divergence, and the level of noise perturbation is further reduced through periodic averaging of model parameters. [1]

3.2 Introduction

Research on machine learning with differential privacy (DP) is nearing the point of making it practical to build effective models that strongly protect the privacy of individuals in the training set [7–9, 11–17]. The introduction of Rényi differential privacy (RDP) [4] offers the promise of additional significant improvements in accuracy in exchange of a very modest relaxation in privacy guarantees.

In this chapter, we propose a novel Rényi differentially private stochastic gradient descent algorithm for convex empirical risk minimization models that outperforms prior work. This method uses output perturbation (with numerical computation of sensitivity) along with the randomness of mini-batches inside stochastic gradient descent to protect privacy.

Let $D = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_n\}$ be a dataset of n i.i.d. samples drawn from an underlying distribution $\mathcal{D}$. The training of many models, including logistic regression and support vector machines, can be reduced to solving an equation such as:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\Theta} F(\mathbf{w}, D) := \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{w}, \mathbf{d}_i)\,,$$

where f measures the *loss* of the model with parameter vector $\mathbf{w}$ on one data point $\mathbf{d}_i$. This procedure is known as Empirical Risk Minimization (ERM). In many cases, f is convex and twice differentiable with respect to model $\mathbf{w}$ [7, 8].

With privacy, the goal is to return a noisy version of the optimal parameter vector $\mathbf{w}^*$, where the noise distribution must satisfy (Rényi) differential privacy constraints [4] to ensure the confidentiality of every record in the training data.

Our solution has the following key features. First, it uses a variation of mini-batch permuting stochastic gradient descent (SGD). This choice provides a computational speedup over full-batch methods [8, 17] and also has implications for privacy: For a large enough dataset, SGD can train a model with many fewer

[1] This chapter is a slightly modified version of [10] published in AISTATS 2019 Proceedings and has been reproduced here, complying with author rights.

passes over the data than full-batch gradient descent and so each data point has a smaller influence on the optimal parameter vector.

Second, we use output perturbation (e.g. where additional noise is added to the optimal parameter vector [8, 9]) instead of gradient perturbation (e.g. where additional noise is added to each mini-batch gradient [13, 14, 17, 18]). This choice allows us to perform periodic averaging of the intermediate parameter vectors encountered in SGD. In convex ERM problems, averaging helps improving convergence (intuitively by reducing the oscillations around the optimum) [19]. Crucially, we show that averaging also helps reducing noise injection to protect privacy. Furthermore, when the input data are initially permuted, the SGD updates are a random combination of many contraction mappings and one expansion mapping. This random behavior allows us to further reduce the amount of noise that is necessary to protect privacy.

By working in the framework of Rényi differential privacy, we are providing a stronger privacy guarantee than (ϵ, δ)-differential privacy yet can still compare against algorithms for (ϵ, δ)-DP using RDP to DP conversion theorems [4]. The contributions in this chapter are summarized as follows:

- We propose a private algorithm based on SGD and output perturbation, for solving convex and smooth empirical risk minimization problems under Rényi differential privacy.

- The inherit randomness in SGD means that the influence of one data point on a parameter update is a random variable. We take advantage of this randomness to reduce the amount of noise that is necessary for protecting privacy with output perturbation.

- Typically, SGD makes slow progress as it approaches the optimum. To counter this effect, we add periodic averaging steps to reduce the oscillations of SGD and to further reduce the amount of noise that is necessary for output perturbation.

- We perform extensive experiment on real datasets against other recently proposed algorithms. We empirically show the effectiveness of the proposed algorithm for a wide range of privacy parameter values.

The rest of this chapter is organized as follows. In Section 3.3, we review the related work. In Section 3.4, we provide background on differential privacy and convex ERM. We present our algorithm in Section 3.5 and experiments in Section 3.6. Section 3.7 concludes this chapter.

3.3 Related Work

The problem of differentially private optimization has received a great deal of attention from the machine learning community, especially in the context of ERM [7–9,11–18,20]. There has been significant advances in both theoretic and algorithmic development.

Bassily et al. [13] derived the minimax lower bound for excess empirical risk and proved that their private variant of SGD matches the lower bound. They used a *gradient perturbation* approach in which the algorithm optimizes the objective function only using gradients perturbed by random noise. However, their private SGD algorithm requires prohibitively large, $O(n^2)$, number of iterations. In [15], Wang et al. also take the gradient perturbation approach to privatize an accelerated version of SGD, called SVRG [21]. They showed that their algorithm also matches the lower bound with improved gradient complexity. Talwar et al. [22] further refined the bound for a specific task by adding more restrictive constraints. Abadi et al. [14] showed that private SGD can be successfully used for deep learning using techniques that would later be known as Rényi differential privacy [4]. Feldman et al. [23] analyzed the privacy amplification effect of noisy contractive mapping in Stochastic Gradient Langevin Dynamics (SGLD), which perform SGD updates using Gaussian noise perturbed gradients. A technique called PATE [16] dramatically improved the private training of deep networks but requries a large public dataset. Using concentrated differential privacy (CDP) [3], Lee and Kifer showed that private full gradient based algorithms can be improved in practice with step size selection and careful allocation of privacy budgets between iterations [17].

The works most similar to ours are that of Wu et al. [9] and that of Zhang et al. [8]. The former proposed an *output perturbation* [2, 7, 8] based SGD algorithm in which noise is calibrated according to the stability analysis of gradient descent. The use of randomly permuted mini-batches in their algorithm is for utility only, whereas we analyze how permutations affect privacy cost (i.e., how they compose with output perturbation and require less additive noise). The latter also used the output perturbation to privatize full batch gradient descent. Output perturbation approaches must derive bounds on the sensitivity of the underlying non-private algorithm and the added noise scales with this bound. The calculation of sensitivity in these prior works often use loose inequalities, resulting in higher noise levels than are necessary. We use numerical computation of sensitivity to substantially decrease the amount of noise added to protect privacy.

Table 3.1: Summary of symbol definitions

Symbol	Definition	Symbol	Definition
D	dataset	B	mini-batch of data
$\mathbf{d}$	a datum record	m	number of mini-batches
$\mathcal{D}$	data universe	i, j	iterate through data records/mini-batches
$\mathbf{x}$	feature record of a datum	η	step size
$\mathcal{X}$	feature universe	s	epoch number
y	target record of a datum	T	total epochs
$\mathcal{Y}$	target universe	t, k	iteration number
n	data size	α	order of Rényi divergence and RDP
$\mathbf{w}$	model vector	ϵ	privacy budget of DP or RDP
Θ	model universe	δ	privacy parameter of approximate DP
F	objective function of D	γ	Gaussian noise sample
f	objective function of d	σ^2	variance of Gaussian noise
$\mathbb{I}$	identity matrix	Δ	sensitivity
ρ	contraction coefficient	R	bound of gradient norm
μ	parameter of strong convexity	λ	regularization coefficient
L	parameter of smoothness	τ	averaging interval

Chaudhuri et al. [7] proposed a general framework called *objective perturbation* as an alternative to gradient and output perturbation: They produce a random objective function by adding a linear noise term to the original objective. To ensure privacy, the resulting problem must be fully solved to optimally using an off-the-shelf optimizer. Kifer et al. [12] further improved the utility of the objective perturbation method by using (ϵ, δ)-DP. Again, the optimization must be solved to optimality (whereas in practice, most optimizers stop with an approximate solution).

3.4 Preliminaries

A record $\mathbf{d}_i \in \mathcal{D} = \mathcal{X} \times \mathcal{Y}$ is a tuple (observation) $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathcal{X}$ is a feature vector and $y_i \in \mathcal{Y}$ is a target (label). A dataset $D \in \mathcal{D}^n$ is a set of n records $\{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_n\}$. Two datasets D and D' are called *neighboring*, denoted as $D \sim D'$, if D' can be obtained from D by replacing one record with another one from $\mathcal{D}$ In this chapter, $\|\cdot\|$ denotes the L_2 norm of a vector. A summary of symbols used in this chapter is presented in Table 3.1.

3.4.1 Convex Empirical Risk Minimization

Recall that a record $\mathbf{d}_i$ consists of a feature vector $\mathbf{x}_i$ and a target y_i. In the empirical risk minimization framework, fitting many convex models can be rephrased as optimizing the following equation.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\Theta} F(\mathbf{w}, D) := \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{w}, \mathbf{x}_i, y_i)\,, \tag{3.1}$$

where Θ is a convex parameter space. F is called the *objective function* defined on the dataset and f is known as the *loss function* of one record (while the terms "objective function" and "loss function" are used interchangeably in some literature). For example a regularized logistic regression model can be fit to a dataset by solving $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\log(1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i)}) + \lambda\|\mathbf{w}\|_2^2)$, in which case the function f equals $\log(1+e^{-y_i(\mathbf{w}\cdot\mathbf{x}_i)}) + \lambda\|\mathbf{w}\|_2^2$. Following prior work on private convex ERM (e.g. [7, 8, 17]), we impose the following conditions on f.

1. **Differentiability.** Loss function f is continuously differentiable with respect to $\mathbf{w}$ and has a continuous Hessian except on a set of Lebesgue measure 0.

2. L**-smooth.** There exists an $L > 0$ such that for any $\mathbf{w}, \mathbf{w}' \in \Theta$ and for all $\mathbf{d}_i = (\mathbf{x}_i, y_i) \in \mathcal{X}\times\mathcal{Y}$,

$$\|\nabla f(\mathbf{w}, \mathbf{d}_i) - \nabla f(\mathbf{w}', \mathbf{d}_i)\|_2 \le L\|\mathbf{w} - \mathbf{w}'\|_2\,.$$

3. **Strong convexity.** There exists a $\mu > 0$ such that $f(\mathbf{w}, \mathbf{d}_i)$ is μ-strongly convex in the first argument, i.e., for any $\mathbf{w}, \mathbf{w}' \in \Theta$ and any $\mathbf{d}_i$,

$$f(\mathbf{w}', \mathbf{d}_i) \ge f(\mathbf{w}, \mathbf{d}_i) + \nabla f(\mathbf{w}, \mathbf{d}_i)^T(\mathbf{w}' - \mathbf{w}) + \frac{\mu}{2}\|\mathbf{w} - \mathbf{w}'\|_2^2\,.$$

4. **Bounded Gradient.** There exists an $R > 0$ that satisfies $\|\nabla f(\mathbf{w}, d_i)\| \le R$ for $\forall \mathbf{w} \in \Theta, d_i \in \mathcal{D}$.

Conditions 1, 2, and 3 imply bounds on the eigenvalues of the Hessian [24], i.e. for $\forall \mathbf{w} \in \Theta$ and $\mathbf{d}_i \in \mathcal{D}$,

$$\mu \mathbb{I} \preccurlyeq \nabla^2 f(\mathbf{w}, \mathbf{d}_i) \preccurlyeq L \mathbb{I} \,. \tag{3.2}$$

except on a set of Lebesgue measure 0. Typically the bounded gradient condition is satisfied by ensuring that the feature vectors $\mathbf{x}_i$ lies inside a ball of some radius R'.

3.4.2 Gradient Descent Operator

Empirical risk minimization, especially in large-scale problems, is often solved with stochastic gradient descent (SGD). At each iteration, SGD chooses a random subset of training data B, called a mini-batch. It uses the gradient on the mini-batch to approximate the full-data gradient of F in (3.1) then updates the parameter vector $\mathbf{w}$ as follows:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \frac{\eta_k}{|B|} \sum_{i \in B} \nabla f(\mathbf{w}_k, \mathbf{d}_i) \,, \tag{3.3}$$

where $\eta_k > 0$ is the step size (which slowly decreases with iteration k). We use the notation ∇f_B to denote the mini-batch gradient for B, i.e., $\nabla f_B(\cdot) = \frac{1}{|B|} \sum_{i \in B} \nabla f(\cdot, \mathbf{d}_i)$. The SGD update in (3.3) can be viewed as an operator on the parameter space Θ defined as

$$\mathcal{T}_B(\cdot) = \mathsf{Id}(\cdot) - \eta \nabla f_B(\cdot) \,, \tag{3.4}$$

where Id is the identity operator. When f_B is smooth and strongly convex, $\mathcal{T}_B$ forms a contraction mapping [25], which means that applying $\mathcal{T}_B$ to any two parameter vectors $\mathbf{w}, \mathbf{w}' \in \Theta$ (using the same mini-batch B) shrinks the distance between $\mathbf{w}$ and $\mathbf{w}'$ by a constant factor. Mathematically, for any $\mathbf{w}, \mathbf{w}' \in \Theta$, we have $\|\mathcal{T}_B(\mathbf{w}) - \mathcal{T}_B(\mathbf{w}')\| \leq \rho \|\mathbf{w} - \mathbf{w}'\|$, where $\rho < 1$ is a contraction coefficient. Note that in (Rényi) differential privacy, a pair of neighboring databases D and D' might produce slightly different mini-batches B and B', which will require us to compare $\mathcal{T}_B(\mathbf{w})$ with $\mathcal{T}_{B'}(\mathbf{w}')$.

3.5 Algorithms

3.5.1 Vanilla Mini-batch SGD

We begin the description of our private SGD algorithm starting with a plain version shown in Algorithm 3.1 which, modulo a few small details, is equivalent to the bold-on method of [9]. We will then add in optimizations that improve performance, resulting in our proposed method, shown in Algorithm 3.3.

Algorithm 3.1 is a non-randomized SGD method. We will later add randomness into the mini-batch construction by permutating record order. Please note that the randomness we sue is different from competing private approaches. For instance, Abadi et al. [14] base their privacy analysis on the assumption that each mini-batch is an independent random sample of the data (also known as Poisson subsampling). Such random sampling is a slow operation. Hence practical implementations do not use this approach. Instead they permute the data and form disjoint mini-batches by taking consecutive records from the permuted data (the resulting mini-batches are therefore not independent since they cannot overlap). Thus we base our privacy analysis on the assumption that the data have been permuted in order to create disjoint mini-batches.

Algorithm 3.1 Vanilla Stochastic Gradient Descent Algorithm (NSGD)

Input: dataset D, number of epochs T, initial step size η_0, number of mini-batches m, noise scale parameter σ^2
Split D into mini-batches $B_0, B_1, \cdots, B_{m-1}$
Initialize $\mathbf{w}_0$
$t \leftarrow 0$
for $s = 1, 2, \cdots, T$ **do**
    $\eta \leftarrow \eta_0/s$
    for $j = 0, 1, \cdots, m-1$ **do**
        $t \leftarrow t + 1$
        $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \eta \frac{1}{|B_j|} \sum_{i \in B_j} \nabla f(\mathbf{w}_{t-1}, d_i)$
    end for
end for
Sample $\gamma \sim \mathcal{N}(0, \sigma^2)$
Output: $\mathbf{w}_{priv} = \mathbf{w}_t + \gamma$

Contraction Mapping. Let us first analyze Algorithm 3.1. It is similar to standard (mini-batched) SGD in that it updates parameters by iteratively applying the SGD operator $\mathcal{T}_{B_i}$ to the current iterate,

i.e., $\mathbf{w}_{t+1} \leftarrow \mathcal{T}_{B_i}(\mathbf{w}_t)$. One important difference is that, while standard SGD constructs the i^{th} batch B_i by randomly selecting the records in the batch, NSGD builds the B_i by splitting D into m mini-batches, denoted by $B_0, B_1, \cdots, B_{m-1}$, and then accesses them sequentially in a cyclic order. This type of algorithm is known as the incremental gradient method [26] and has been widely used to solve large-scale problems.

This sequential access has important implications for our sensitivity analysis. Let D and D' be neighboring datasets. If D is the input to the algorithm, then the algorithm will split it into mini-batches $B_0, B_1, \cdots, B_{m-1}$ and will iteratively produce a sequence of parameter estimates $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \cdots$, where $\mathbf{w}_0$ is some initial starting value, and for $t > 0$, the model updates are performed using the SGD rule $\mathbf{w}_t \leftarrow \mathcal{T}_{B_{t-1 \mod m}}(\mathbf{w}_{t-1})$. On the other hand, if D' is the input, the algorithm will split it into mini-batches $B'_0, B'_1, \cdots, B'_{m-1}$ and produce a sequence of parameter estimates $\mathbf{w}'_0, \mathbf{w}'_1, \mathbf{w}'_2, \cdots$, where $\mathbf{w}_0 = \mathbf{w}'_0$ is the default starting value and the model updates use the rule $\mathbf{w}'_t \leftarrow \mathcal{T}_{B'_{t-1 \mod m}}(\mathbf{w}'_{t-1})$.

Following [9], note that since D and D' differ on the value of one record, there will be a j such that mini-batches B_j and B'_j differ on the value of one record. For all $i \neq j, i = 0, ..., m-1$, we have $B_i = B'_i$. This means that as long as $(t \mod m) \neq j$, $\mathbf{w}_{t+1}$ and $\mathbf{w}'_{t+1}$ are obtained using the same operator $\mathcal{T}_{B_i}$ (i.e., $\mathbf{w}_{t+1} \leftarrow \mathcal{T}_{B_i}(\mathbf{w}_t)$ and $\mathbf{w}'_{t+1} \leftarrow \mathcal{T}_{B_i}(\mathbf{w}'_t)$). Now, when the loss function f satisfies condition 1, 2, and 3 in Section 3.4.2, operator $\mathcal{T}_{B_i}$ is a contraction, so that the parameter estimates move closer to each other. That is, let $\Delta_{t+1} := \|\mathbf{w}_{t+1} - \mathbf{w}'_{t+1}\|$ and $\Delta_t := \|\mathbf{w}_t - \mathbf{w}'_t\|$, then the contraction property means that $\Delta_{t+1} < \Delta_t$.

However, when $(t \mod m) = j$, the parameter updates use different operators: $\mathcal{T}_{B_j}$ and $\mathcal{T}_{B'_j}$ so $\mathbf{w}_{t+1} \leftarrow \mathcal{T}_{B_i}(\mathbf{w}_t)$ and $\mathbf{w}'_{t+1} \leftarrow \mathcal{T}_{B'_i}(\mathbf{w}'_t)$. In this situation, the operators force the parameter estimates further apart. This discussion is summarized in the following lemma.

Lemma 3.1. *Let B and B' be mini-batches that differ on the value of one record. Define the operator $\mathcal{T}_B(\cdot) := \mathit{Id}(\cdot) - \eta \nabla f_B(\cdot)$ (and similarly for B'). Let $\mathbf{w}$ and $\mathbf{w}'$ be any two vectors in Θ. Let $\rho = \max\{|1 - \eta\mu|, |1 - \eta L|\}$ (where μ is the strong convexity parameter and L is the smoothness parameter). Then:*

$$\|\mathcal{T}_B(\mathbf{w}) - \mathcal{T}_B(\mathbf{w})\| \leq \rho \|\mathbf{w} - \mathbf{w}'\| \tag{3.5}$$

$$\|\mathcal{T}_B(\mathbf{w}) - \mathcal{T}_{B'}(\mathbf{w})\| \leq \rho \|\mathbf{w} - \mathbf{w}'\| + \frac{2\eta R}{|B|} \tag{3.6}$$

where the first equation shows the effect of using the same operator $\mathcal{T}_B$ and the second equation shows the effect of using $\mathcal{T}_B$ to update $\mathbf{w}$ and a different operator $\mathcal{T}_{B'}$ to update $\mathbf{w}'$.

Proof. We first consider the case where the same operator $\mathcal{T}_B$ is applied to both $\mathbf{w}$ and $\mathbf{w}'$, i.e., $B = B'$.

$$
\begin{aligned}
\|\mathcal{T}_B(\mathbf{w}) - \mathcal{T}_B(\mathbf{w}')\| &= \|\mathbf{w} - \eta\nabla f_B(\mathbf{w}) - (\mathbf{w}' - \eta\nabla f_B(\mathbf{w}'))\| \\
&= \|\mathbf{w} - \mathbf{w}' - \eta(\nabla f_B(\mathbf{w}) - \nabla f_B(\mathbf{w}'))\| \\
&= \left\| \int_0^1 \{\mathbb{I} - \eta\nabla^2 f_B(\mathbf{w}' + s(\mathbf{w} - \mathbf{w}'))\}(\mathbf{w} - \mathbf{w}')\, \mathrm{d}s \right\| \\
&\leq \int_0^1 \left\| \{\mathbb{I} - \eta\nabla^2 f_B(\mathbf{w}' + s(\mathbf{w} - \mathbf{w}'))\}(\mathbf{w} - \mathbf{w}') \right\| \mathrm{d}s \\
&\leq \int_0^1 \left\| \{\mathbb{I} - \eta\nabla^2 f_B(\mathbf{w}' + s(\mathbf{w} - \mathbf{w}'))\} \right\| \|(\mathbf{w} - \mathbf{w}')\|\, \mathrm{d}s \\
&\leq \int_0^1 \sup_{\mathbf{z}} \|\mathbb{I} - \eta\nabla^2 f_B(\mathbf{z})\| \|(\mathbf{w} - \mathbf{w}')\|\, \mathrm{d}s \\
&\leq \sup_{\mathbf{z}} \|\mathbb{I} - \eta\nabla^2 f_B(\mathbf{z})\| \|(\mathbf{w} - \mathbf{w}')\| \\
&\leq \max\{|1 - \eta\mu|, |1 - \eta L|\} \|(\mathbf{w} - \mathbf{w}')\|
\end{aligned}
$$

where $\mathbf{z} = \mathbf{w}' + s^*(\mathbf{w} - \mathbf{w}')$, $s^* \in [0, 1]$ is a point on the line segment joining $\mathbf{w}$ and $\mathbf{w}'$.

Now we consider the case where B and B' differ by one record. Let ξ denote the index of record at which D and D' differ, i.e., $d_i = d'_i$ for all $i \neq \xi$, and $d_\xi \neq d'_\xi$. We introduce the following equality.

$$
\begin{aligned}
\nabla f_B(\mathbf{w}) - \nabla f_{B'}(\mathbf{w}') =& \frac{1}{|B|}\left\{ \sum_{i \in B} \nabla f(\mathbf{w}, \mathbf{d}_i) - \sum_{i \in B'} \nabla f(\mathbf{w}', \mathbf{d}'_i) \right\} \\
=& \frac{1}{|B|}\big\{ \nabla f(\mathbf{w}, \mathbf{d}_\xi) - \nabla f(\mathbf{w}', \mathbf{d}_\xi) + \nabla f(\mathbf{w}', \mathbf{d}_\xi) - \nabla f(\mathbf{w}', \mathbf{d}'_\xi) \\
&+ \sum_{i \in B, i \neq \xi} \nabla f(\mathbf{w}, \mathbf{d}_i) - \nabla f(\mathbf{w}', \mathbf{d}_i) \big\} \\
=& \frac{1}{|B|}\left\{ \big(\nabla f(\mathbf{w}', \mathbf{d}_\xi) - \nabla f(\mathbf{w}', \mathbf{d}'_\xi)\big) + \sum_{i \in B} \nabla f(\mathbf{w}, \mathbf{d}_i) - \nabla f(\mathbf{w}', \mathbf{d}_i) \right\} \\
=& \nabla f_B(\mathbf{w}) - \nabla f_B(\mathbf{w}') + \frac{1}{|B|}\big(\nabla f(\mathbf{w}', \mathbf{d}_\xi) - \nabla f(\mathbf{w}', \mathbf{d}'_\xi)\big)
\end{aligned}
$$

Using this equation, we get

$$\begin{aligned}
\|\mathcal{T}_B(\mathbf{w}) - \mathcal{T}_{B'}(\mathbf{w}')\| =& \|\mathbf{w} - \eta\nabla f_B(\mathbf{w}) - (\mathbf{w}' - \eta\nabla f_{B'}(\mathbf{w}'))\| \\
=& \|\mathbf{w} - \mathbf{w}' - \eta(\nabla f_B(\mathbf{w}) - \nabla f_{B'}(\mathbf{w}'))\| \\
=& \left\|\mathbf{w} - \mathbf{w}' - \eta(\nabla f_B(\mathbf{w}) - \nabla f_B(\mathbf{w}')) - \frac{\eta}{|B|}\big(\nabla f(\mathbf{w}', d_\xi) - \nabla f(\mathbf{w}', \mathbf{d}'_\xi)\big)\right\| \\
\leq& \|\mathbf{w} - \mathbf{w}' - \eta(\nabla f_B(\mathbf{w}) - \nabla f_B(\mathbf{w}'))\| + \frac{\eta}{|B|}\|\nabla f(\mathbf{w}', \mathbf{d}_\xi) - \nabla f(\mathbf{w}', \mathbf{d}'_\xi)\| \\
\leq& \|\mathbf{w} - \mathbf{w}' - \eta(\nabla f_B(\mathbf{w}) - \nabla f_B(\mathbf{w}'))\| + \frac{2\eta R}{|B|} \\
=& \|\mathcal{T}_B(\mathbf{w}) - \mathcal{T}_B(\mathbf{w}')\| + \frac{2\eta R}{|B|} \\
\leq& \rho\|\mathbf{w} - \mathbf{w}'\| + \frac{2\eta R}{|B|}
\end{aligned}$$

where the second to last inequality is due to our requirement on the boundedness of gradient. □

If we use a fixed step size η, instead of diminishing step size, in Algorithm 3.1 (i.e. removing line 6) and set $m = 1$, then the naive algorithm coincides with private full-batch gradient descent (GD) proposed in [8]. The following Theorem shows how the sensitivity of the resulting algorithm can be computed using the recurrence relation from Lemma 3.1.

Theorem 3.1. *If we run Algorithm 3.1 for arbitrary number of epochs with a fixed step size η, its sensitivity Δ satisfies*

$$\Delta \leq \frac{2\eta R}{|B|(1-\rho^m)} \tag{3.7}$$

where $\rho = \max\{|1-\eta\mu|, |1-\eta L|\}$. In particular, when $m = 1$ and $\eta = \frac{2}{L+\mu}$, $\Delta \leq \frac{2R}{n\mu}$.

Proof. Let D and D' be any two databases that differ on one record. Given a fixed randomness in data permutation, let $B_0, \cdots, B_{m-1}$ and $B'_0, \cdots, B'_{m-1}$ denote m disjoint mini-batches for D and D', respectively. Then there exists an index j such that $B_j \neq B'_j$ and $B_i = B'_i$ for all $i \neq j$.

Algorithm 3.1 on input D generates a sequence of solutions $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \cdots$, using the rule $\mathbf{w}_i \leftarrow \mathcal{T}_{B_{i-1 \bmod m}}(\mathbf{w}_{i-1})$ (and similarly on input D' using $\mathcal{T}_{B'}$. Define $\Delta^{(k)}$ as the difference between $\mathbf{w}_i$ and $\mathbf{w}'_i$ at the end of k^{th} epoch. Provided that the algorithm for input D and D' starts with the same initial solution, i.e., $\mathbf{w}_0 = \mathbf{w}'_0$, Lemma 3.1 shows that the first $j-1$ updates in an epoch will be contractions, the j^{th} updates will be an expansion, and the remaining $m-j$ updates will be contractions. Therefore,

at the end of the first epoch, we have $\Delta^{(1)} \leq \rho^{m-j}\frac{2\eta R}{|B|}$. In the second epoch, there will be again $j-1$ contractions, one expansion, and $m-j$ contractions. Hence, we have

$$\Delta^{(2)} \leq \rho^{m-j}\left(\rho(\rho^{j-1}\Delta^{(1)}) + \frac{2\eta R}{|B|}\right) = \rho^m \cdot \rho^{m-j}\frac{2\eta R}{|B|} + \rho^{m-j}\frac{2\eta R}{|B|}\,.$$

Likewise, at the end of the k^{th} epoch,

$$\Delta^{(k)} \leq \rho^{m-j}\frac{2\eta R}{|B|}\left(\rho^{(k-1)m} + \rho^{(k-2)m} + \cdots + \rho^m + 1\right)$$

Therefore,

$$\lim_{k\to\infty}\Delta^{(k)} = \frac{\rho^{m-j}2\eta R}{|B|(1-\rho^m)} \leq \frac{2\eta R}{|B|(1-\rho^m)}$$

since $0 < \rho < 1$. Since $\rho = \max\{|1-\eta\mu|, |1-\eta L|\}$ is a function of η, the value of η can be optimized to minimize ρ (i.e. to obtain the maximum contraction). It can be calculated that ρ has the minimum value of $\frac{L-\mu}{L+\mu}$ when $\eta = \frac{2}{L+\mu}$, which is when $|1-\eta\mu| = |1-\eta L|$. Plugging $\rho = \frac{L-\mu}{L+\mu}$ and $m = 1$ into (3.7) results in the second claim. □

Note that the above sensitivity is tighter than the one computed in [8], which is $\frac{5R(L+\mu)}{n\mu L}$.

Sensitivity Calculation. With a fixed number of epochs and a step size η_t that depends on the epoch number s, we can use Lemma 3.1 as follows. If D and D' only differ in the last mini-batch, then their initial parameters are the same. After the first epoch, they differ by at most $\frac{2\eta_1 R}{|B|}$, after then second, they differ by at most $\rho^m\frac{2\eta_1 R}{|B|} + \frac{2\eta_2 R}{|B|}$, and so on. It is easy to see that it is the worst case (upper bound is maximized), when D and D' differ on the last of the m mini-batches, and hence this upper bound (which can be computed numerically) is the sensitivity. After the last epoch, noise with variance $\sigma^2 = \alpha\Delta^2/(2\epsilon)$ can be added to the final parameter vector to achieve (α, ϵ)-RDP.

3.5.2 Randomized Permuted Mini-batches

If we permute the data *once* before starting the SGD updates, then the mini-batches become randomized and are subsequently processed in cyclic order. This allows us to add less noise because, with randomized mini-batches, it is no longer possible to create "bad" neighbors that always differ in the last batch: Now the mini-batch they differ in will be $0, 1, ..., m-1$ with equal probability $\frac{1}{m}$.

Algorithm 3.2 Sensitivity and Privacy Calculation

Input: total number of epochs T, initial step size η_0, number of mini-batches m, mini-batch size $|B|$, upper bound of gradient norm R, probability vector $\mathbf{q}$, noise scale σ^2.
function ComputeSensitivity($T, m, \eta_0, |B|, R$)
    Initialize $\Delta[:]$
    for $j = 1, \cdots, m$ **do**
        $\Delta[j] \leftarrow 0$ ▷ initialization
    end for
    for $s = 1, \cdots, T$ **do**
        $\eta \leftarrow \eta_0/s$
        $\rho \leftarrow \max\{|1 - \eta\mu|, |1 - \eta L|\}$
        for $j = 1, \cdots, m$ **do**
            $\Delta[:] \leftarrow \rho\Delta[:]$ ▷ contraction
            $\Delta[j] \leftarrow \Delta[j] + \frac{2\eta R}{|B|}$ ▷ expansion
        end for
    end for
    Output: Δ
end function
function ComputePrivacy($\alpha, \Delta, \mathbf{q}, \sigma^2$)
    $H_\alpha \leftarrow 0$
    for $j = 1, \cdots, m$ **do**
      $H_\alpha \leftarrow H_\alpha + \mathbf{q}[j] \times \exp(\alpha(\alpha - 1)\Delta[j]^2/(2\sigma^2))$
    end for
    Output: $\epsilon = \log(H_\alpha)/(\alpha - 1)$
end function

Let D and D' be *any* two databases that differ by one record. Consider what happens when the algorithm is run with input D (world 1) and with input D' (world 2). Given the same input randomness, when the data are split into mini-batches, for any $j = 1, ..., m$, with probability $\frac{1}{m}$ world 1 and world 2 will differ in the j^{th} mini-batch only. So, starting with the same initial model weight vector $\mathbf{w}_0$, the first $j - 1$ model updates in an epoch will be contractions, the j^{th} update will be an expansion, and the remaining $m - j$ will be contractions again. Hence, after the first epoch, the difference in weights between the two worlds is at most $\Delta^{(1)} = \rho^{m-j} 2\eta_1 R/|B|$; and after the second epoch, the difference is bounded by $\Delta^{(2)} = \rho^{m-j}(\rho^j \Delta^{(1)} + 2\eta_2 R/|B|)$, etc. In the end, the algorithm adds Gaussian noise to the vector.

This scenario can be abstracted as follows. There exists a set of mechanisms[2] $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_m$ with associated L_2 sensitivities $\Delta_1, \Delta_2, \cdots, \Delta_m$ and an associated probability vector $\mathbf{q} = [q_1, q_2, \cdots, q_m]$ (which adds up to 1). An algorithm $\mathcal{M}$, on input D, randomly samples an index i 1, ..., m according to $\mathbf{q}$, and returns $\mathcal{M}_i(D)$. An upper bound on the privacy cost of $\mathcal{M}$ is given by Lemma 3.2.

Lemma 3.2. *Define* $H_\alpha(P_1; P_2) := e^{(\alpha-1)\mathbb{D}_\alpha(P_1 \| P_2)}$. *Let* $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_m$ *be mechanisms and* $\mathbf{q} = [q_1, q_2, \cdots, q_m]$ *be a probability vector over* $[1, 2, \cdots, m]$. *Let* $\mathcal{M}$, *on input* D, *sample* $i \sim \mathbf{q}$ *and return* $\mathcal{M}_i(D)$, *and let* $D' \sim D$. *Then*

$$H_\alpha(\mathcal{M}(D); \mathcal{M}(D')) \leq \sum_{j=1}^{m} q_j H_\alpha(\mathcal{M}_j(D); \mathcal{M}_j(D'))\,.$$

Proof. For each j, let P_1^j and P_2^j be the distributions of $\mathcal{M}_j(D_1)$ and $\mathcal{M}_j(D_2)$, respectively (D_1 and D_2 can be neighboring). Let P_1 be the distribution of $\mathcal{M}(D_1)$ and P_2 be the distribution of $\mathcal{M}(D_2)$.

$$\begin{aligned}
H_\alpha(\mathcal{M}(D_1); \mathcal{M}(D_2)) &= \mathbb{E}_{x\sim P_2}[P_1(x)^\alpha P_2(x)^{-\alpha}] \\
&= \mathbb{E}_{x\sim P_2}\big[\big(\sum_{j=1}^{m} q_j P_1^j(x) / \sum_{j=1}^{m} q_j P_2^j(x)\big)^\alpha\big] \\
&= \mathbb{E}_{x\sim P_2}\left[\left(\sum_{j=1}^{m} \frac{q_j P_2^j(x)}{\sum_{j'=1}^{m} q_{j'} P_2^{j'}(x)} \frac{P_1^j(x)}{P_2^j(x)}\right)^\alpha\right] \\
&= \mathbb{E}_{x\sim P_2}\left[\left(\sum_{j=1}^{m} \frac{q_j P_2^j(x)}{P_2(x)} \frac{P_1^j(x)}{P_2^j(x)}\right)^\alpha\right] \\
&\leq \mathbb{E}_{x\sim P_2}\left[\sum_{j=1}^{m} \frac{q_j P_2^j(x)}{P_2(x)} \left(\frac{P_1^j(x)}{P_2^j(x)}\right)^\alpha\right] \\
&= \sum_{j=1}^{m} q_j E_{x\sim P_2^j}\left[\big(P_1^j(x)\; P_2^j(x)\big)^\alpha\right] \\
&= \sum_{j=1}^{m} q_j H_\alpha\big(\mathcal{M}_j(D_1); \mathcal{M}_j(D_2)\big)
\end{aligned}$$

where the inequality comes from Jensen's inequality (since the function $z \to z^\alpha$ is convex for $\alpha > 1$ and the second to last equality comes from using the definition of expected value. □

[2] For example, $\mathcal{M}_i$ can apply a function f_i on the input data and then add $\mathcal{N}(0, \sigma^2\mathbb{I})$ noise to the answer.

The final privacy cost of $\mathcal{M}$ can be obtained by noting $\mathbb{D}_\alpha(P_1 \| P_2) = \log(H_\alpha(P_1; P_2))/(\alpha - 1)$. The function ComputeSensitivity in Algorithm 3.2 combines Lemma 3.1 and 3.2 to compute the Δ's corresponding to the version of Algorithm 3.1 in which data records are firstly randomly permuted.

3.5.3 SGD with Averaging

One of the drawbacks of SGD is its progress slows as it approaches the optimum (due to the requirement for diminishing step size [27], it has sub-linear convergence rate even when the objective function is strongly convex). To alleviate this problem, every τ epochs, *RSGD-AR* performs averaging of parameters [28, 29] (line 15) over the most recent τ epochs. It then resets the step size (line 16), thus emulating restarts. Instead of starting from scratch, the algorithm uses the averaged value of last $m\tau$ iterates as the starting value of the restart [30]. It is known that both averaging and warm restarts help improve convergence properties of stochastic approximation algorithms. The full algorithm, with these enhancements, is described in Algorithm 3.3 as RSGD-AR.

Algorithm 3.3 Revised Stochastic Gradient Descent with Averaging (RSGD-AR)

1: **Input**: dataset D, number of epochs T, initial step size η_0, averaging interval τ, number of mini-batches m, noise scale parameter σ^2
2: Randomly permute the dataset D
3: Construct mini-batches $B_0, B_1, \cdots, B_{m-1}$
4: Initialize $\mathbf{w}_0$
5: $t \leftarrow 0, h \leftarrow 0$
6: **for** $s = 1, 2, \cdots, T$ **do**
7: $\quad h \leftarrow h + 1$ ▷ epoch count of each run
8: $\quad \eta \leftarrow \eta_0 / s$
9: $\quad$ **for** $j = 0, 1, \cdots, m-1$ **do**
10: $\quad\quad t \leftarrow t + 1$
11: $\quad\quad \mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \eta \frac{1}{|B_j|} \sum_{i \in B_j} \nabla f(\mathbf{w}_{t-1}, d_i)$
12: $\quad$ **end for**
13: $\quad$ **if** $s \mod \tau = 0$ **then**
14: $\quad\quad t \leftarrow t + 1$
15: $\quad\quad \mathbf{w}_t \leftarrow \frac{1}{m\tau} \sum_{k=1}^{m\tau} \mathbf{w}_{t-k}$ ▷ averaging
16: $\quad\quad h \leftarrow 0$ ▷ step size reset
17: $\quad$ **end if**
18: **end for**
19: Sample $\gamma \sim \mathcal{N}(0, \sigma^2)$
20: **Output**: $\mathbf{w}_{priv} = \mathbf{w}_t + \gamma$

Let $T_j = j\tau$ denote an epoch at which the j^{th} averaging is performed (so it is update number $jm\tau+j$).

$$\|\mathbf{w}_{jm\tau+j} - \mathbf{w}'_{jm\tau+j}\| = \left\| \frac{1}{m\tau} \sum_{k=(j-1)m\tau+j}^{jm\tau+j-1} \mathbf{w}_k - \mathbf{w}'_k \right\| \leq \frac{1}{m\tau} \sum_{k=(j-1)m\tau+j}^{jm\tau+j-1} \|\mathbf{w}_k - \mathbf{w}'_k\| \tag{3.8}$$

Thus, for a given permutation of the data, we can average the upper bound on parameter differences before applying Lemma 3.2. That is, every τ epochs, we average the most recent $m\tau$ values of Δ values in Algorithm 3.2. Putting it all together:

Theorem 3.2. *Algorithm 3.3 with averaging satisfies* (α, ϵ)*-RDP, where*

$$\epsilon = \frac{1}{\alpha - 1} \log \left(\frac{1}{m} \sum_{j=1}^{m} e^{\frac{\alpha(\alpha-1)(\Delta[j])^2}{2\sigma^2}} \right). \tag{3.9}$$

Proof. Let D and D' be neighboring datasets. Let $\mathcal{M}_j$ be a mechanism with associated sensitivity $\Delta[j]$. Given the randomly permuted input dataset, Algorithm 3.3, denoted by $\mathcal{M}$, chooses $\mathcal{M}_j$ with probability $q_j = 1/m$ and releases the output using the Gaussian mechanism with noise scale parameter σ^2. We show that the Rényi divergence between the output distributions of $\mathcal{M}$ is bounded by ϵ.

$$\begin{aligned}
\mathbb{D}_\alpha(\mathcal{M}(D) \| \mathcal{M}(D')) &= \frac{1}{\alpha - 1} \log H_\alpha(\mathcal{M}(D); \mathcal{M}(D')) \\
&\leq \frac{1}{\alpha - 1} \log \left(\sum_{j=1}^{m} q_j H_\alpha(\mathcal{M}_j(D); \mathcal{M}_j(D')) \right) \\
&= \frac{1}{\alpha - 1} \log \left(\frac{1}{m} \sum_{j=1}^{m} e^{(\alpha-1)\mathbb{D}_\alpha(\mathcal{M}_j(D) \| \mathcal{M}_j(D'))} \right) \\
&\leq \frac{1}{\alpha - 1} \log \left(\frac{1}{m} \sum_{j=1}^{m} e^{\alpha(\alpha-1)\Delta[j]^2/2\sigma^2} \right),
\end{aligned}$$

where the first and second inequalities are due to Lemmas 3.2 and 2.1, respectively. □

The function ComputePrivacy in Algorithm 3.2 computes the privacy parameter ϵ based on Δ's at the end of training, $\mathbf{q} = [\frac{1}{m}, \cdots, \frac{1}{m}]$, and the noisy scale σ^2.

Table 3.2: Summary of datasets

Dataset	Size	Dimension
Adult	$48,842$	124
Bank	$45,211$	33
IPUMS-BR	$38,000$	53
IPUMS-US	$40,000$	58
KDDCup99	$4,898,431$	120

3.6 Experimental Results

This section shows the experimental set-up and results of proposed algorithm, comparing with several private baselines.

3.6.1 Datasets

We evaluate the performance of the proposed algorithm using 5 real datasets: (1) Adult [31, 32] data extracted from the 1994 US Census Data. (2) BANK [32] data about marketing campaigns of a finance institution. (3) IPUMS-BR and (4) IPUMS-US data extracted from IPUMS data [33] (5) KDDCup99 data [34] collected from a simulated network. Table 3.2 shows the number of records and number of attributes (after pre-processing).

3.6.2 Preprocessing

We performed the following standard preprocessing operations. Every categorical attribute is converted into a set of binary variables using one-hot encoding. For each unique category, a new binary attribute is created. All numerical attributes are re-scaled into the range $[0, 1]$ to ensure that they have the same scale. Additionally, for those methods that require feature vectors to lie inside a bounded space, we normalize each observation to a unit norm (i.e., $\|x\|_2 = 1$ for $i = 1, 2, \cdots, n$).

3.6.3 Baselines

We compare the proposed algorithm, RSGD-AR, against seven baseline algorithms, [3] namely, ObjPert [7,12], OutPert-GD [8], DP-AGD [17], SGD-MA [14], NSGD (described below), NonPrivate, and Majority. ObjPert is an objective perturbation method that optimizes the objective function pertured with random noise. OutPert-GD is an output perturbation method that injects Gaussian noise to the solution obtained by running GD with a fixed step size. DP-AGD is a gradient perturbation method that uses carefully chhosen step sizes with adaptive privacy budget. SGD-MA is also a gradient perturbation based SGD algorithm that uses an improved composition method, called moment accountant. NSGD takes the Vanilla algorithm (essentially equivalent to [9]) but uses numerical sensitivity calculations to reduce added noise. NonPrivate optimizes the objective with L-BFGS [35] and does not add noise to achieve privacy. Majority predicts the most frequent label. We repeat 5-fold cross validation 20 times and report average classification accuracy and final objective value.

3.6.4 Hyper-parameter settings

Throughout all the experiments, the value of privacy parameter δ is fixed to 10^{-8} for the Adult, Bank, IPUMS-US, and IPUMS-BR datasets and to 10^{-12} for the KDDCup99 dataset. The mini-batch size is fixed to $4,000$ for RSGD-AR and $\sqrt{n}$ for SGD-MA, where n is the size of dataset.

3.6.5 Logistic Regression and SVM

We compare performance on two different tasks: regularized logistic regression (LR) and support vector machine (SVM). For logistic regression, we define

$$f(\mathbf{w}, \mathbf{x}_i, y_i) = \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 , \tag{3.10}$$

where $\mathbf{x}_i \in \mathbb{R}^{p+1}$, $y_i \in \{-1, +1\}$, and $\lambda > 0$ is a regularization coefficient. The value of λ is fixed to 0.001 in all experiments. Figure 3.1 shows the classification accuracy (top) and final objective values (bottom) of algorithms on the four datasets. For all the values of ϵ tested, the proposed RSGD-AR

[3] We omit a comparison to the noisy SLGD in [23] because, due to its large per-iteration noise requirement, for small values of δ and datasets used in our experiments we observed it diverges unless ϵ is very large.

algorithm consistently outperforms or performs competitively with other baselines. In all figures, the accuracy of SGD-MA suddenly surges to almost that of non-private algorithm at a certain value of ϵ (because, for each privacy budget it can handle, we ran it with multiple choices of pre-specified number of iterations, to tune its performance).

For SVM, in order to satisfy the differentiability conditions in Section 3.4.1, we use the huberized hinge loss function [7, 36] defined as

$$f(\mathbf{w}, \mathbf{x}_i, y_i) = \ell_{\mathsf{huber}}(y\mathbf{w}^T\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2\,, \tag{3.11}$$

and the unregularized hinge loss

$$\ell_{\mathsf{huber}}(z) := \begin{cases} 0 & \text{if } z > 1+h \\ \frac{1}{4h}(1+h-z)^2 & \text{if } |1-z| \leq h \\ 1-z & \text{if } z < 1-h \end{cases} \tag{3.12}$$

where h is a hyperparameter. In our experiments, we fixed $h = 0.5$. The performance of the proposed algorithm on SVM task is shown in Figure 3.2. As it is shown, RSGD-AR outperforms or achieves similar performance with other baseline algorithms.

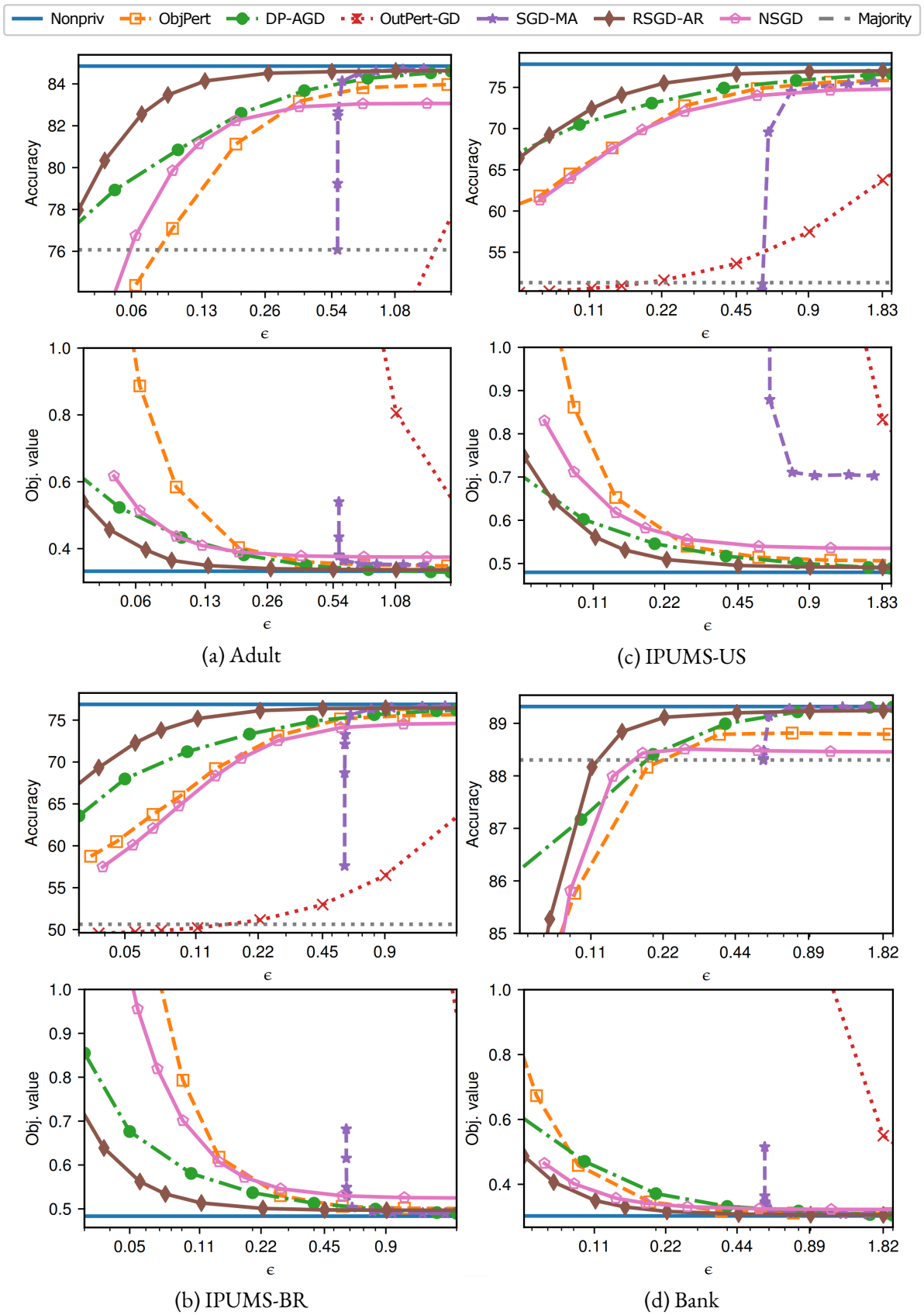
3.6.6 Performance and Processing Time on Large Dataset

We evaluate the proposed algorithm on KDDCup99 to demonstrate the performance on a large dataset. Figure 3.3 hows the performance of LR and SVM. For LR, output perturbation methods perform better when ϵ is small while gradient perturbation methods outperform when ϵ is large. This is because the sensitivity is inversely proportional to the dataset size to output perturbation methods.

To compare the speed of algorithms, we generated 5 sub-samples of size $50K, 100K, 150K, 200K,$ and $250K$ from KDDCup99 dataset,and measured each algorithm's processing time on them. As shown in Figure 3.4, ObjPert which uses L-BFGS to solve the perturbed problem is the fastest. It is observed that L-BFGS finds an approximate solution with reasonable accuracy on KDDCup99 dataset in less than 20 iterations. While our method requires more iterations, it is almost as fast as the ObjPert algorithm.

3.7 Conclusions

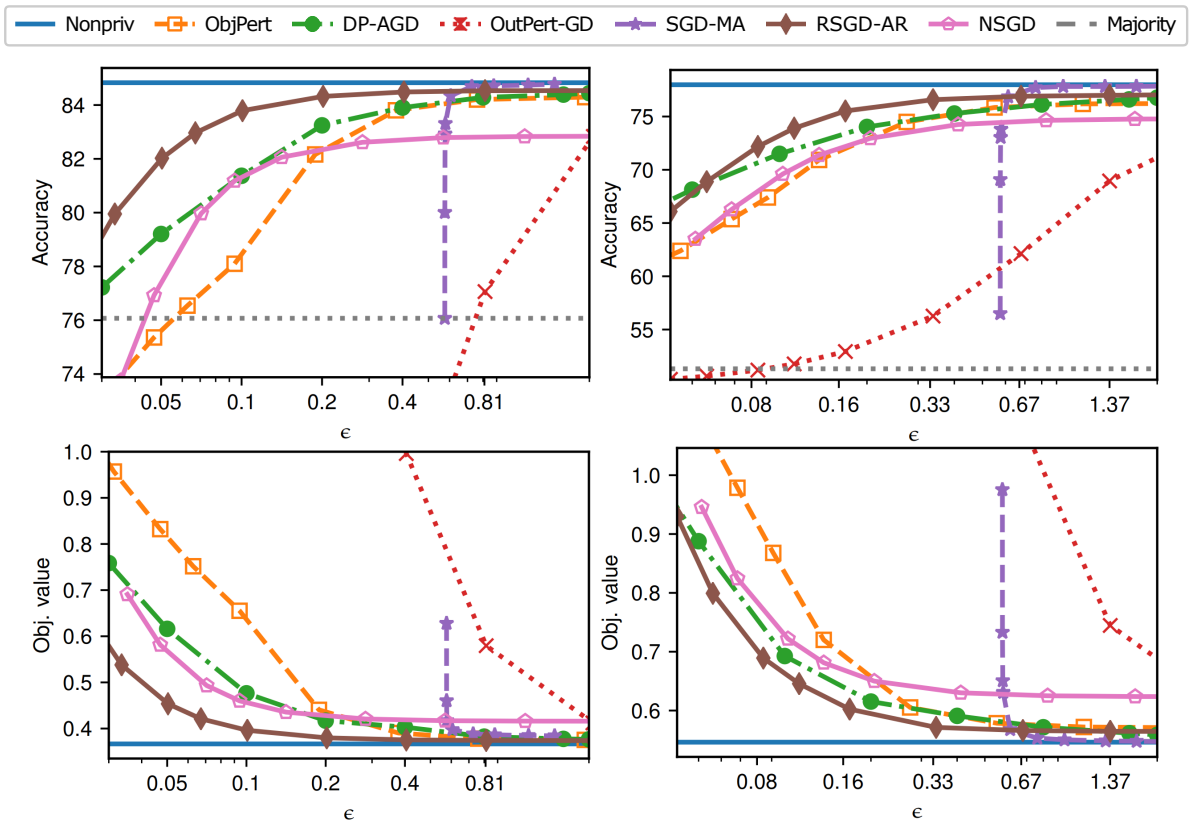
We presented an SGD algorithm for Rényi differentially private convex empirical risk minimization. It is based on output perturbation, which allows it to take advantage of an averaging technique for accelerating SGD, and also accounts for the added privacy caused by batch randomization. This algorithm is suitable for large scale problems and experimentally outperformed prior (ϵ, δ)-differentially private algorithm.



(a) Adult (c) IPUMS-US

(b) IPUMS-BR (d) Bank

Figure 3.1: Logistic regression by varying ϵ (Top: Classification accuracies; Bottom: Objective values)



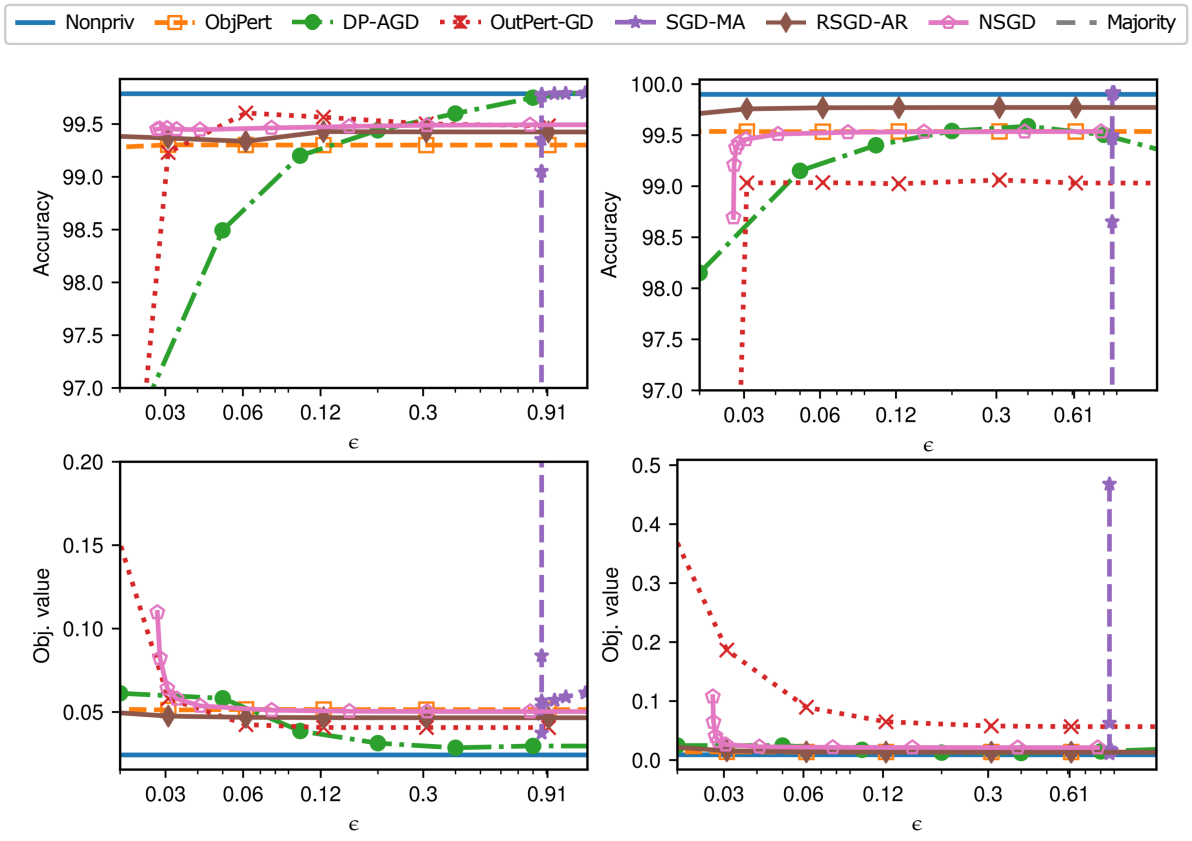
(a) Adult

(c) IPUMS-US

(b) IPUMS-BR

(d) Bank

Figure 3.2: SVM by varying ϵ (Top: Classification accuracies; Bottom: Objective values)



(a) Logistic Regression (b) SVM

Figure 3.3: Performance on KDDCup99 dataset (Left: LR, Right: SVM)

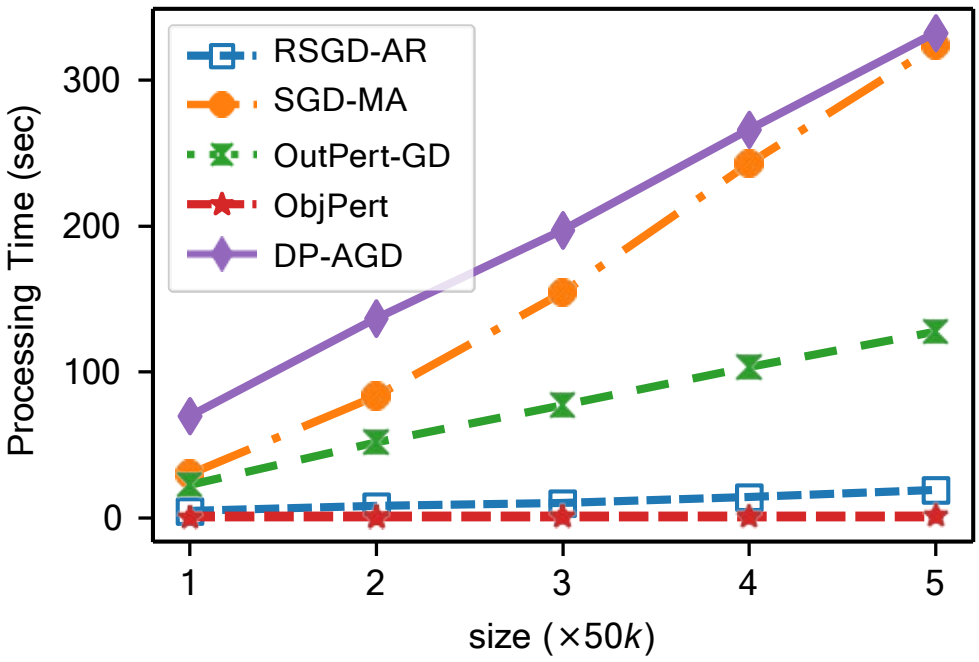


Figure 3.4: Processing times for 5 different subsamples of KDDCup99 dataset

Chapter 4

Rényi Differentially Private ADMM for Non-Smooth Regularized Optimization

4.1 Motivation

One drawback of the RSGD-AR algorithm proposed in Chapter 3 is that it requires the smoothness of the objective function, including the regularization portion. Such restriction is held for many other output and objective perturbation algorithms, such as [8, 12]. However, there exist many non-smooth regularizers popular in application. For example, L_1 regularized logistic regression is famous for learning on sparse data and feature selection. Despite the non-differentiability of non-smooth regularization, most gradient perturbation methods can still be used on these problems, but perturbations have to be applied on proximal gradients, where the performance can be adversely affected with the existence of noise. In this chapter, a novel private ERM algorithm is presented, where the data-dependent model training and the data-independent regularization are separated by the stochastic ADMM algorithm, and performed in an augmented manner. By this separation, closed form solutions can be obtained for both portions, and proximal gradients for the non-differentiable part can be avoided. [1]

[1] This chapter is a slightly modified version of [37] published in ACM CODASPY 2020 Proceedings and has been reproduced here, complying with author rights.

4.2 Introduction

Concerns on privacy of individuals in the data used for training machine learning models have led to extensive research on private model building techniques [7, 8, 10, 12–15], especially in the context of Empirical Risk Minimization (ERM). Let $D = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n\}$ be a dataset, where $\mathbf{d}_i \in \mathcal{D}$ for $i = 1, ..., n$. Many machine learning problems can be formulated as regularized optimization problems of the form:

$$\min_{\mathbf{x}\in\Theta} F(\mathbf{x}, D) := \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}, \mathbf{d}_i) + \lambda h(\mathbf{x}) \tag{4.1}$$

where $\lambda > 0$ is a regularization coefficient, $f : \Theta \times \mathcal{D} \to \mathbb{R}$ is a smooth convex objective function, and $h : \Theta \to \mathbb{R}$ is a simple convex *non-smooth* regularizer such as L_1-norm or nuclear norm. This formulation has received substantial attention as it arises in many interesting applications of machine learning such as generalized LASSO [38], matrix recovery [39, 40], and a class of L_1 regularized problems. Despite recent advances in methods for differentially private ERM, many existing solutions are not directly applicable to the problem in (4.1) due to requirement for differentiability [8, 10, 13, 14] or strong convexity and smoothness [7] of the regularization term $h(\mathbf{x})$. Alternating direction method of multipliers (ADMM) [41] has shown to be effective in solving optimization problems with complicated structural regularization.

In this chapter, we propose two stochastic ADMM algorithms that satisfy Rényi Differential Privacy (RDP), namely subsampled stochastic ADMM (ssADMM) and model perturbation based ADMM (mpADMM). The first algorithm has the following key features. First, ssADMM is scalable and fast. The algorithm splits the composite objective function into differentiable and non-smooth terms, $\sum_i f(\mathbf{x}, \mathbf{d}_i)$ and $h(\mathbf{x})$, using the ADMM framework. The differentiable term is further approximated by the first order Taylor expansion and linearization as in [42]. This approximated augmented Lagrangian function has a simple analytical solution. For the non-smooth regularization term $h(\mathbf{x})$, ssADMM applies proximal mappings. For many non-smooth regularization function popularly used in machine learning, such as L_1-norm, SCAD [43], and MCP [44], those proximal mappings yield closed form solutions. Therefore, both subproblems can be solved efficiently.

Second, ssADMM makes use of the recently proposed *privacy amplification* lemma [5] to tightly bound the total privacy loss across many iterations. In the closed-form solution of the modified augmented Lagrangian function, the only data dependent term is the gradient calculation $\nabla f(\mathbf{x}^k)$, where $\mathbf{x}^k$ denotes

the value of $\mathbf{x}$ at iteration k. The algorithm computes the gradient $\nabla f(\mathbf{x}^k)$ using a randomly *subsampled* data and add Gaussian noise to ensure (α, ϵ_k)-RDP, which allows us to exploit the randomness in the subsampling and to introduce less noise to each iteration to achieve a certain privacy level.

The second algorithm, mpADMM, takes the output perturbation approach but substantially differs from the original method. Unlike the original method which releases model parameters once only at the end, the proposed method releases the output after each epoch. For each epoch, we numerically compute the sensitivity of both primal and dual variable updates in ADMM and release the parameter vector using the Gaussian mechanism. The algorithm uses the released (noisy) output as the starting value for the next epoch.

Our contributions are summarized as follows:

- We propose two efficient Rényi differentially private algorithms, based on stochastic ADMM, for solving non-smooth convex optimization problems. In our proposed ssADMM, each subproblem is solved exactly in closed form.

- We apply the recent privacy amplification result for RDP to stochastic ADMM and show that the inherent randomness in subsampling process can be used to achieve stronger privacy protection.

- We empirically show the effectiveness of the proposed algorithms by performing extensive empirical evaluations on generalized linear models and comparing with other baseline algorithms. The results show that, in high privacy regimes (small ϵ), ssADMM and mpADMM outperform other baseline algorithms in terms of classification and feature selection performance, respectively.

The rest of this chapter is organized as follows: Section 4.3 summarizes related work. In Section 4.4, we provide background on Rényi differential privacy and ADMM. Section 4.5 introduces the proposed Rényi differentially private ADMM algorithms. Section 4.6 provides the performance evaluations on both synthetic and real datasets. Section 4.7 concludes the chapter.

4.3 Related Work

Many works have been done to solve the empirical risk minimization (ERM) problem under differential privacy. Generally, there are three types of algorithms proposed. Output perturbation algorithms perturb the model parameters based on *sensitivity*, for example, [7] analyzed the sensitivity of optimal solutions

trained between neighboring databases; [8] tackled the case when full gradient descent is applied; and [9] and [10] analyzed the situation of applying stochastic gradient descent on permuting mini-batches. Objective perturbation algorithms perturb the training *objective functions*, and the privacy guarantee is subject to an exact solution of the ERM problem: [7] presented the first objective perturbation technique and showed its advantage to the naive output perturbation algorithm. It is extended and improved by [12], which applies Gaussian mechanism and achieves (ϵ, δ)-DP. Gradient perturbation algorithms perturb the (stochastic) gradients used for model updating by first-order optimization methods, and use a composition technique to quantify the overall privacy leak for multiple access of the data through gradient calculation. For example, [13] proposed "strong composition" theorem, then [14] proposed "moment accountant" method, which is also used in [15] and [45]. The Réyni differential privacy was introduced by [4], which can also be applied in gradient perturbation, especially after [5] proposed its amplification by subsampling results.

Alternating Direction Method of Multipliers (ADMM) is an old algorithm to solve optimization problems [46]. It has been extensively studied, and applied in many domains such as outlier recovery [47], image processing [48], and sensor detection [49]. In addition to its original version, many variations have been presented, such as [42, 50, 51]. Several ADMM based differentially private algorithms have been presented, for example, [52] applied objective perturbation technique on the original ADMM problem, [53] and [54] applied output and objective perturbation technique, and [55] applied gradient perturbation technique on ADMM-based algorithms in distributed settings.

L_1 regularized ERM problem was first incorporated into ERM for linear regression, that is least absolute shrinkage and selection operator (LASSO) [56]. Some variants of LASSO exists, such as [57] and [58]. It has also been used for classification problems, and many algorithms for solving L_1 regularized generalized linear models (GLMs) were presented, such as [59–61]. [62] and [63] has shown that L_1 regularized classification has good performance in feature selection. L_1 regularization has been occasionally used in neural networks, but [64] has shown that L_1 regularized multi-layer perception model is improperly learnable in polynomial time. Nevertheless, it is popularly among GLMs, which is a group of models widely used in industrial domains like finance and education, where many sensitive data are involved. However, limited to the assumption on the loss function, many differentially private ERM algorithms cannot be directly applied on L_1 regularized classification, with a few exceptions such as [14, 52, 55].

Table 4.1: Summary of symbol definitions

Symbol	Definition	Symbol	Definition
D	dataset	p	data dimension
$\mathcal{D}$	data distribution	$\mathbf{x}$	primal variable vector of f
$\mathbf{d}$	a datum	$\mathbf{z}$	primal variable vector of h
n	data size	$\mathbf{y}$	dual variable vector
F	loss function over dataset	$\mathbf{g}$	gradient vector
ℓ	loss function over one datum	$\mathcal{S}$	soft-thresholding operator
f	unregularized objective function	$\mathbb{I}$	identity matrix
h	regularization function	α	order of Rényi divergence and RDP
i	iterate through a dataset	γ	Gaussian noise sample
j	iterate through a vector	σ^2	variance of Gaussian noise
k	iteration number of model training	ϵ	privacy budget of DP or RDP
T	total iteration steps	δ	privacy parameter of approximate DP
B	subsampled mini-catch	Θ	model space
m	mini-batch size	η	learning rate
$\mathbf{s}$	feature vector of one datum	ρ	penalty parameter for ADMM
l	label of one datum	λ	regularization coefficient

4.4 Preliminaries

In this section we introduce relative background of this chapter. We will start with definitions and lemmas in differential privacy and Rényi differential privacy, the L_1-regularized classification problem we aim to solve, and then the ADMM algorithm based on which we proposed our algorithms.

We assume a dataset $D = \{\mathbf{d}_1, \mathbf{d}_1, \ldots, \mathbf{d}_n\} \sim \mathcal{D}^n$ is a dataset collected from n individuals from an unknown population distribution $\mathcal{D}$, where $\mathbf{d}_i = (\mathbf{s}_i, l_i)$ for $i = 1, ..., n$ is a record of one individual, with $\mathbf{s}_i$ being a vector of features of dimension p, and $l_i \in \{-1, +1\}$ being its label. Two datasets D and D' are considered *neighboring*, if D' can be obtained by replacing one record with another one from $\mathcal{D}$, notated as $D \sim D'$. We use bold lowercase letters to denote vectors, and plain lowercase letters to denote scalars. We use $\|\cdot\|_1$ (resp. $\|\cdot\|_2$) as L_1 (resp. L_2) norm of a vector. A summary of symbol notations (except some temporarily defined symbols in proofs, which is otherwise specified) in this chapter is presented in Table 4.1.

4.4.1 Regularized Empirical Risk Minimization

Many problems in machine learning can be formulated as empirical risk minimization (ERM), which seek a solution $x^* \in \Theta$ that minimizes an empirical loss on the training data:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Theta} F(\mathbf{x}, D) := \arg\min_{\mathbf{x}\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}, \mathbf{d}_i)\,,$$

where Θ is a model parameter space, ℓ is a *loss* function. To prevent overfitting, it is common to add a (data-independent) regularization term into the objective function to form the loss function, i.e. $\ell(\mathbf{x}, \mathbf{d}_i) = f(\mathbf{x}, \mathbf{d}_i) + R(\mathbf{x})$. For example, L_1 regularized logistic regression, where $R(\mathbf{x}) = \lambda h(\mathbf{x}) := \lambda\|\mathbf{x}\|_1$, one can fit the model by solving

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-l_i\mathbf{x}^T\mathbf{s}_i)) + \lambda\|\mathbf{x}\|_1\,, \tag{4.2}$$

and λ is a hyperparameter called regularization coefficient. Recall that each datum $\mathbf{d}_i = (\mathbf{s}_i, l_i)$ as a tuple of feature vector $\mathbf{s}_i$ and class label l_i. However, due to that many optimization algorithms assume the loss function to be doubly differentiable, it cannot be directly used on L_1 regularization problems. In this chapter, we make the following assumptions on the loss function:

- **Convexity** Both the data-dependent function f and regularization function h are convex.
- **Differentiability** The non-regularized data-dependent function f is continuously differentiable with respect to $\mathbf{x}$.
- **Bounded gradient** There exists a constant $C > 0$ such that $\|\nabla f(\mathbf{x}, \mathbf{d})\|_2 \leq C$ for all $\mathbf{x} \in \Theta$ and $\mathbf{d} \in \mathcal{D}$. Usually it is satisfied by preprocessing the data to ensure the feature $\mathbf{s}_i$ of each data $\mathbf{d}_i$ lies inside a ball of some radius r, or directly clip the L_2 norm of individual gradient by a threshold C.

4.4.2 Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) algorithm was proposed decades ago, and has recently been widely used to solve optimization problems in machine learning [46]. Consider the

optimization problem [2]

$$\begin{aligned} \text{minimize} \quad & f(x) + h(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^m \to \mathbb{R}$, $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, and $c \in \mathbb{R}^p$. ADMM forms the augmented Lagrangian of the problem:

$$L_\rho(x, z, y) := f(x) + h(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \tag{4.3}$$

where x, z are called the *primal* variables, $y \in \mathbb{R}^p$ is called the *dual* variable, and $\rho > 0$ is a pre-selected *penalty* parameter.

ADMM algorithm solves the optimization problem by alternating the iterations below [46]

$$x\text{-minimization step: } x^{k+1} \leftarrow \arg\min_x L_\rho(x, z^k, y^k) \tag{4.4}$$

$$z\text{-minimization step: } z^{k+1} \leftarrow \arg\min_z L_\rho(x^{k+1}, z, y^k) \tag{4.5}$$

$$\text{dual variable update: } y^{k+1} \leftarrow y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \tag{4.6}$$

Therefore, x and z are updated in an *alternating* fashion, and separating minimization over x and z into two steps can make the otherwise hard-to-solve optimization problem solvable in a sequential manner.

4.4.3 Stochastic ADMM

One variant of ADMM, stochastic ADMM (sADMM), was proposed by [42] and tested on L_1 regularized linear regression (LASSO). This variant was proposed based on the observation that, for ADMM problems, usually one of $f(x)$ and $h(z)$ is data-dependent, and it is both expensive and unnecessary to exactly solve its minimization step for each iteration. To be specific, let f be data-dependent, and h be data-independent, then the optimization problem becomes $f(x, D) + h(z)$, and sADMM approximate

[2] In section 4.4.2 and 4.4.3, f, h represent any functions, x, y, z, c represent any vectors, and A, B represent any matrices.

L_ρ by *approximated* augmented Lagrangian $\hat{L}_\rho$, defined at iteration k as

$$\begin{aligned}\hat{L}_\rho(x,z,y) := f(x^k) + \langle \nabla f(x^k, B_k), x\rangle + \frac{\|x-x^k\|_2^2}{2\eta^k} \\ +h(z) + y^T(Ax+Bz-c) + \frac{\rho}{2}\|Ax+Bz-c\|_2^2\end{aligned} \tag{4.7}$$

where B_k is a portion of the data accessed at iteration k, and η^k is the learning rate at iteration k. After this approximation of L_ρ by $\hat{L}_\rho$, one can derive an exact solution for each x-minimization step in (4.4), instead of solving a computationally expensive ERM problem.

4.5 Algorithms

In this section we propose the main algorithms. We propose two sADMM based L_1 regularized classification algorithms, both satisfying Rényi differential privacy. One achieves privacy by gradient perturbation relying on randomized subsampling; the other is through model perturbation after each epoch relying on sensitivity calculation. Both algorithms assume a centralized computing: all training data were collected in a center (data curator), which performs all the computation locally. This is because we assume the data is small-to-median sized, where L_1 regularizations are usually applied on.

4.5.1 Rényi differentially private subsampling algorithm

Our subsampling private sADMM algorithm (s s ADMM) is presented in Algorithm 4.1. This algorithm is inspired by the differentially private stochastic gradient descent (DP-SGD) technique proposed in [14].

Similar as DP-SGD, our s s ADMM algorithm perturbs the mini-batch gradient by Gaussian noise right after gradient evaluation in line 6. However, Algorithm 4.1 differs from DP-SGD for the following aspects: (i) By utilizing ADMM, we are able separate gradient descent and L_1 regularization into two steps, so that pure gradient can be computed and perturbed in x-minimization step; for DP-SGD, proximal gradient has to be used to handle L_1 regularization; (ii) while DP-SGD suggest using constant learning rate, we proved that using decreasing step size in Algorithm 4.1 help accelerate convergence, as in Theorem 4.2 and numerical experiments; (iii) authors of DP-SGD proposed the moment accountant (MA) method to analyze the privacy loss, and convert to (ϵ, δ)-DP; we use the most recent RDP for

subsampling mechanism, which is a more advanced technique to analyze privacy loss, and also easier to implement.

Algorithm 4.1 RDP subsampling sADMM L_1 regularized ERM algorithm (s s A D M M)

Input: Dataset $D = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_n\}$. Penalty parameter ρ, mini-batch size m, total iterations T.
Initialize: primal variables $\mathbf{x}^0$, $\mathbf{z}^0$, dual variable $\mathbf{y}^0$.
for iteration $k = 0, 1, ..., T-1$ **do**
    Sample mini-batch B_k from D of size m.
    $\mathbf{g}_k \leftarrow \frac{1}{m} \sum_{\mathbf{d}_i \in B_k} \nabla f(\mathbf{x}^k, \mathbf{d}_i)$ ▷ compute gradient
    $\tilde{\mathbf{g}}_k \leftarrow \mathbf{g}_k + \gamma$ where $\gamma \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_p)$ ▷ perturb gradient by Gaussian noise
    Compute $\mathbf{x}^{k+1}$ by (4.8) using $\tilde{\mathbf{g}}_k$ ▷ primal variable $\mathbf{x}$
    Compute $\mathbf{z}^{k+1}$ by (4.10) ▷ primal variable $\mathbf{z}$
    Compute $\mathbf{y}^{k+1}$ by (4.9) ▷ dual variable $\mathbf{y}$
end for
Output: $\mathbf{x}^T$

Recall the approximated augmented Lagrangian defined in (4.7). For L_1 regularized ERM, let $h(z)$ be the regularization term $R(\mathbf{z}) = \lambda \|\mathbf{z}\|_1$, the constraint $Ax + Bz = c$ reduces to $\mathbf{x} = \mathbf{z}$, then by taking derivative of $\hat{L}_\rho(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k)$ and set to zero, one get

$$\mathbf{x}^{k+1} \leftarrow \frac{1}{\rho + 1/\eta^k}(-\nabla f(\mathbf{x}, B_k) - \mathbf{y}^k + \rho \mathbf{z}^k + \mathbf{x}^k/\eta^k) \tag{4.8}$$

as the exact solution to minimize $\hat{L}_\rho(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k)$, and

$$\mathbf{y}^{k+1} \leftarrow \mathbf{y}^k + \rho(\mathbf{x} - \mathbf{z}) \tag{4.9}$$

to update the dual variable y.

Since the regularization is data-independent, it does not cause any privacy leak. Therefore, any (non-) smooth regularizers are applicable for Algorithm 4.1, with the same privacy guarantee. Since we use L_1 regularization as an example, for the z-minimization step, we utilize soft-thresholding technique from [46]

to acquire the solution to minimize $L_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k)$:

$$\mathbf{z}^{k+1} \leftarrow \mathcal{S}_{\frac{\lambda}{\rho}}(\mathbf{x}^{k+1} + \mathbf{y}^k/\rho) \tag{4.10}$$

where soft-thresholding $\mathcal{S}$ is an element-wise operator defined as

$$\mathcal{S}_t(\mathbf{x})_j = \begin{cases} x_j - t & \text{if } x_j > t \\ x_j + t & \text{if } x_j < -t \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

where x_j is the j-th element of $\mathbf{x}$. Similar technique has been used in [42] and [52].

Another ADMM based algorithm proposed in [55] (DP-ADMM) also used gradient perturbation technique. Our method differed from theirs for the following aspects: (i) DP-ADMM is used for distributed learning, so that the training objective is assigned into multiple parties each holding a portion of the data, instead in ssADMM it is the data dependent loss and regularization that are separated; (ii) in DP-ADMM, each party is perturbing full gradient and transmit to the center, so that there is no privacy amplification effect, therefore although both algorithms solve optimization approximately, their privacy cost is higher than ours at each step. Our methods differ from the ADMM-OBJP method (DPLL in [52]) for the following aspect: (i) ADMM-OBJP perturb the training objective at each iteration, and use full gradient descent multiple times to acquire exact solution at each iteration, which is not as efficient as ours, since our method only access a portion of data once at each step; (ii) ADMM-OBJP guarantees privacy only if exact solution is acquired at each step. Since almost all optimization algorithms are approximate (like gradient descent), their privacy guarantee is only theoretically true. The privacy guarantee of ssADMM is given by Theorem 4.1.

Theorem 4.1. *Algorithm 4.1 is* (α, ϵ)*-RDP.*

Proof. We first show the L_2 sensitivity of mini-batch gradient $\mathbf{g}_k$. Assume neighboring mini-batches B_k and B'_k differ by one record $\mathbf{d}_s \in B_k$ and $\mathbf{d}'_s \in B'_k$, by Definition 2.4,

$$\begin{aligned}\Delta_2(\mathbf{g}_k) =& \Delta_2[\frac{1}{m}\sum_{\mathbf{d}_i \in \mathbf{B}_k} \nabla f(\mathbf{x}^k, \mathbf{d}_i)] \\ =& \sup_{B_k \sim B'_k} \|\frac{1}{m}\sum_{\mathbf{d}_i \in B_k} \nabla f(x^k, d_i) - \frac{1}{m}\sum_{\mathbf{d}_i \in B'_k} \nabla f(\mathbf{x}^k, \mathbf{d}_i)\|_2 \\ =& \frac{1}{m}\sup \|\nabla f(\mathbf{x}^k, \mathbf{d}_s) - \nabla f(\mathbf{x}^k, \mathbf{d}'_s)\|_2 \leq \frac{2C}{m}\end{aligned}$$

Let $\epsilon_k(\alpha) = \alpha(\Delta_2^2(g_k))/2\sigma^2$. So each iteration is $(\alpha, \epsilon_k(\alpha))$-RDP by Lemma 2.1, with respect to the batch B_k. Since B_k is a randomized subsample of D, by Lemma 2.3, we can calculate $\epsilon'_k(\alpha)$ so that each iteration is $(\alpha, \epsilon'_k(\alpha))$-RDP with respect to D. Since the algorithm has run T iterations, let $\epsilon = \sum_{k=0}^{T-1} \epsilon'_k(\alpha)$, by Lemma 2.2, Algorithm 4.1 is (α, ϵ)-RDP. □

Theorem 4.2. *If we choose $\eta^k = O(1/\sqrt{k})$, and train for T iterations, then Algorithm 4.1 has the expected convergence rate of $O(1/\sqrt{T})$.*

Proof. The proof is done by applying similar technique for Theorem 1 in [42], considering the Gaussian noise term added. Define

$$\mathbf{u} := \begin{pmatrix}\mathbf{x} \\ \mathbf{z}\end{pmatrix}, \overline{\mathbf{u}}^k := \begin{pmatrix}\frac{1}{k}\sum_{i=1}^{k-1}\mathbf{x}^i \\ \frac{1}{k}\sum_{i=1}^{k-1}\mathbf{z}^i\end{pmatrix}, \theta(\mathbf{u}) := f(\mathbf{x}) + h(\mathbf{z}),$$

and define

$$\mathbf{w} := \begin{pmatrix}\mathbf{x} \\ \mathbf{z} \\ \mathbf{y}\end{pmatrix}, \overline{\mathbf{w}}^k := \begin{pmatrix}\frac{1}{k}\sum_{i=1}^{k-1}\mathbf{x}^i \\ \frac{1}{k}\sum_{i=1}^{k-1}\mathbf{z}^i \\ \frac{1}{k}\sum_{i=1}^{k-1}\mathbf{y}^i\end{pmatrix}, F(\mathbf{w}) := \begin{pmatrix}-\mathbf{y} \\ \mathbf{y} \\ \mathbf{x} - \mathbf{z}\end{pmatrix}$$

Denote $\mathbf{u}^* := \begin{pmatrix}\mathbf{x}^* \\ \mathbf{z}^*\end{pmatrix}$ as the optimal solution, and $\delta_{k+1} := \nabla f(\mathbf{x}^k, B_k) - \nabla f(\mathbf{x}^k, D)$, $d_{\mathcal{X}} := \sup_{\mathbf{x}_a, \mathbf{x}_b \in \mathcal{X}} \|\mathbf{x}_a - \mathbf{x}_b\|$, $d_{\mathbf{y}^*} := \|\mathbf{y}^0 - \mathbf{y}^*\|$.

Therefore, consider the expectation of $\theta(\overline{\mathbf{u}}^t) - \theta(\mathbf{u}^*)$ after t iterations,

$$\mathbb{E}\left[\theta(\overline{\mathbf{u}}^t) - \theta(\mathbf{u}^*) + (\overline{\mathbf{w}}^t - \mathbf{w}^*)^T F(\overline{\mathbf{w}}^t)\right]$$

$$
\begin{aligned}
=&\mathbb{E}\left[\theta(\overline{\mathbf{u}}^t)-\theta(\mathbf{u}^*)+(\overline{\mathbf{x}}^t-\mathbf{x}^*)^T(-\overline{\mathbf{y}}^t)+(\overline{\mathbf{z}}^t-\mathbf{z}^*)^T(\overline{\mathbf{y}}^t)+(\overline{\mathbf{y}}-\mathbf{y})^T(\overline{\mathbf{x}}^t-\overline{\mathbf{z}}^t)\right]\\
\leq&\mathbb{E}\left[\frac{1}{t}\sum_{k=0}^{t-1}\big[\frac{\eta^k}{2}\|\nabla f(\mathbf{x}^k,B_k)+\gamma^k\|^2+\frac{1}{2\eta^k}(\|\mathbf{x}^k-\mathbf{x}^*\|^2-\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2)\right.\\
&\left.+\langle\delta_{k+1},\mathbf{x}^*-\mathbf{x}^k\rangle\big]+\frac{1}{t}\big(\frac{\rho}{2}\|\mathbf{x}^*-\mathbf{z}^0\|^2+\frac{1}{2\rho}\|\mathbf{y}-\mathbf{y}^0\|^2\big)\right]\\
\leq&\mathbb{E}\left[\frac{1}{t}\sum_{k=0}^{t-1}\big[\frac{\eta^k(C^2+p\sigma^2)}{2}+\langle\delta_{k+1},\mathbf{x}^*-\mathbf{x}^k\rangle\big]+\frac{1}{t}\big(\frac{d_{\mathcal{X}}^2}{2\eta^{t-1}}+\frac{\rho}{2}d_{\mathbf{y}^*}^2+\frac{1}{2\rho}\|\mathbf{y}-\mathbf{y}^0\|^2\big)\right]\\
=&\mathbb{E}\left[\frac{1}{t}\sum_{k=0}^{t-1}\big[\frac{\eta^k(C^2+p\sigma^2)}{2}\big]+\frac{1}{t}\big(\frac{d_{\mathcal{X}}^2}{2\eta^{t-1}}+\frac{\rho}{2}d_{\mathbf{y}^*}^2+\frac{1}{2\rho}\|\mathbf{y}-\mathbf{y}^0\|^2\big)\right].
\end{aligned}
\tag{4.12}
$$

The first inequality holds by applying an expected version of Lemma 2 in [42], note that since noisy perturbation $\gamma\sim\mathcal{N}(0,\sigma^2\mathbb{I}_p)$, $\mathbb{E}[\nabla f(\mathbf{x}^k,B_k)+\gamma]=\nabla f(\mathbf{x}^k,B_k)$, and $\mathbb{E}[\|\nabla f(\mathbf{x}^k,B_k)+\gamma^k\|^2]\leq\mathbb{E}[\|\nabla f(\mathbf{x}^k,B_k)\|^2]+\mathbb{E}[\|\gamma\|^2]+2\mathbb{E}[\|\nabla f(\mathbf{x}^k,B_k)\|]\mathbb{E}[\gamma]\leq C^2+p\sigma^2$.

The last equality holds because we assume $\mathbf{x}^k$ is independent of B_k, since B_k is drawn to calculate $\mathbf{x}^{k+1}$. Hence $\mathbb{E}_{B_k|B_{[0:k-1]}}\langle\delta_{k+1},\mathbf{x}^*-\mathbf{x}^k\rangle=0$.

The above holds for all dual variable $\mathbf{y}$, hence it holds for $\mathbf{y}$ in a ball $\mathcal{B}_0=\{\mathbf{y}:\|\mathbf{y}\|_2\leq\beta\}$. According to (33) in [42],

$$
\max_{\mathbf{y}\in\mathcal{B}_0}\{\theta(\overline{\mathbf{u}}^t)-\theta(\mathbf{u}^*)+(\overline{\mathbf{w}}^t-\mathbf{w}^*)^TF(\overline{\mathbf{w}}^t)\}=\theta(\overline{\mathbf{u}}^t)-\theta(\mathbf{u}^*)+\beta\|\overline{\mathbf{x}}_t-\overline{\mathbf{z}}_t\|
$$

Therefore, continue on (4.12), we can have

$$
\begin{aligned}
&\mathbb{E}\big[\theta(\overline{\mathbf{u}}^t)-\theta(\mathbf{u}^*)+\beta\|\overline{\mathbf{x}}_t-\overline{\mathbf{z}}_t\|\big]\\
\leq&\mathbb{E}\left[\frac{1}{t}\sum_{k=0}^{t-1}\big[\frac{\eta^k(C^2+p\sigma^2)}{2}\big]+\frac{1}{t}\big(\frac{d_{\mathcal{X}}^2}{2\eta^{t-1}}+\frac{\rho}{2}d_{\mathbf{y}^*}^2+\frac{1}{2\rho}\|\mathbf{y}-\mathbf{y}^0\|^2\big)\right]\\
\leq&\mathbb{E}\left[\frac{1}{t}\sum_{k=0}^{t-1}\big[\frac{\eta^k(C^2+p\sigma^2)}{2}\big]+\frac{1}{t}\big(\frac{d_{\mathcal{X}}^2}{2\eta^{t-1}}+\frac{\rho}{2}d_{\mathbf{y}^*}^2\big)\right]+\mathbb{E}\left[\max_{y\in\mathcal{B}_0}\{\frac{1}{2\rho t}\|\mathbf{y}-\mathbf{y}_0\|^2\right]\\
\leq&\frac{1}{t}\left(\frac{C^2+p\sigma^2}{2}\sum_{k=1}^{t}\eta^k+\frac{d_{\mathcal{X}}^2}{2\eta^{t-1}}\right)+\frac{\rho d_{\mathbf{y}^*}^2}{2t}+\frac{\beta^2}{2\rho t}
\end{aligned}
$$

So if we choose $\eta^k=\frac{d_{\mathcal{X}}}{\sqrt{2(C^2+p\sigma^2)k}}=O(1/\sqrt{k})$, after t iterations, $\mathbb{E}\big[\theta(\overline{u}^t)-\theta(u^*)+\beta\|\overline{x}_t-\overline{z}_t\|\big]\leq\frac{d_{\mathcal{X}}\sqrt{2(C^2+p\sigma^2)}}{\sqrt{t}}+\frac{\rho d_{\mathbf{y}^*}^2}{2t}+\frac{\beta^2}{2\rho t}=O(1/\sqrt{t})$.

Hence, Algorithm 4.1 has convergence rate of $O(1/\sqrt{T})$. □

4.5.2 Rényi differentially private model perturbation algorithm

Our model perturbation private sADMM algorithm (MPADMM) is presented in Algorithm 4.2. Different from perturbing the gradients, this algorithm use the unperturbed gradients to do model calculation for a whole step, and keep track of the L_2 sensitivity of all data-dependent model vectors. After each epoch, Gaussian noises are injected into model vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, and total privacy ϵ is updated, according to calculated sensitivity and σ^2. Due to it is difficult to calculate the sensitivity over multiple epochs, we perform output perturbation after each epoch. Therefore, this algorithm can be considered as multiple-time output perturbation algorithm.

Algorithm 4.2 RDP model perturbation sADMM L_1 regularized ERM algorithm (MPADMM)

1: **Input**: Dataset $D = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_n\}$. Penalty parameter ρ, total epochs T.
2: **Initialize**: primal variables $\mathbf{x}^0$, $\mathbf{z}^0$, dual variable $\mathbf{y}^0$.
3: **for** epoch $k = 0, 1, ..., T-1$ **do**
4: $\quad \mathbf{g}_k \leftarrow \frac{1}{n}\sum_{\mathbf{d}_i \in D} \nabla f(\mathbf{x}^k, \mathbf{d}_i)$ ▷ compute gradient
5: $\quad$ Compute $\mathbf{x}^{k+1}$ by (4.8) ▷ primal variable $\mathbf{x}$
6: $\quad$ Compute $\mathbf{z}^{k+1}$ by (4.10) ▷ primal variable $\mathbf{z}$
7: $\quad$ Compute $\mathbf{y}^{k+1}$ by (4.9) ▷ dual variable $\mathbf{y}$
8: $\quad$ Sample $\gamma_1, \gamma_2, \gamma_3 \sim N(0, \sigma^2 \mathbb{I}_p)$
9: $\quad \mathbf{x}^{k+1} \leftarrow \mathbf{x}^{k+1} + \gamma_1, \mathbf{y}^{k+1} \leftarrow \mathbf{y}^{k+1} + \gamma_2, \mathbf{z}^{k+1} = \mathbf{z}^{k+1} + \gamma_3$ ▷ perturb the model
10: **end for**
11: **Output**: $\mathbf{x}^T$

To calculate the sensitivity, since unperturbed batch gradient is used here, after one epoch, all primal and dual variables are data-dependent. Assume neighboring datasets D and D' differ at position s: $\mathbf{d}_s \in D$ and $\mathbf{d}'_s \in D'$. We define $\delta_{\mathbf{x}} := \mathbf{x} - (\mathbf{x}')$ where $\mathbf{x}$ and $(\mathbf{x}')$ are primal variables evaluated on D and D',

respectively, after one epoch. Also, define $\delta_{\mathbf{z}}^k$ and $\delta_{\mathbf{y}}^k$ similarly. Then, after epoch k,

$$\begin{aligned}
\delta_{\mathbf{x}}^{k+1} =& \mathbf{x}^{k+1} - (\mathbf{x}')^{k+1} \\
=& \frac{1}{\rho + 1/\eta^k}(-\frac{1}{n}\sum_{\mathbf{d}_i \in D} \nabla f(\mathbf{x}^k, \mathbf{d}_i) - \mathbf{y}^k + \rho \mathbf{z}^k + \mathbf{x}^k/\eta^k) - \\
& \frac{1}{\rho + 1/\eta^k}(-\frac{1}{n}\sum_{\mathbf{d}_i \in D'} \nabla f(\mathbf{x}^k, \mathbf{d}_i) - \mathbf{y}^k + \rho \mathbf{z}^k + \mathbf{x}^k/\eta^k) \\
=& (\nabla f(\mathbf{x}^k, \mathbf{d}'_s) - \nabla f(\mathbf{x}^k, \mathbf{d}_s))/n(1 + \eta^{k+1}\rho)
\end{aligned}$$

Consider when the soft-thresholding operator $\mathcal{S}_t$ (4.11) applied on two vectors $\mathbf{w}$ and $\mathbf{w}'$, and compare $\mathcal{S}_t(\mathbf{w}) - \mathcal{S}_t(\mathbf{w}')$ with $\mathbf{w} - \mathbf{w}'$ element-wise (at positions $j = 1, ..., p$, where p is the dimension of $\mathbf{w}$):

- If w_j and w'_j are of different signs, applying $\mathcal{S}$ on w_j and w'_j would bring them closer, therefore $|\mathcal{S}_t(w_j) - \mathcal{S}_t(w'_j)| < |w_j - w'_j|$;
- If w_j and w'_j are of the same sign, without loss of generality, let $|w_j| \leq |w'_j|$. One can easily observe that
  - If $t \leq |w_j| \leq |w'_j|$, then $|\mathcal{S}_t(w_j) - \mathcal{S}_t(w'_j)| = |(|w_j| - t) - (|w'_j| - t)| = |w_j - w'_j|$;
  - If $|w_j| < t < |w'_j|$, then $|\mathcal{S}_t(w_j) - \mathcal{S}_t(w'_j)| = |0 - (|w'_j| - t)| < |w_j - w'_j|$ since $t < |w'_j|$;
  - If $|w_j| \leq |w'_j| \leq t$, then $|\mathcal{S}_t(w_j) - \mathcal{S}_t(w'_j)| = 0 \leq |w_j - w'_j|$;

For vectors $\mathbf{u}$, $\mathbf{v}$, we can use $\mathbf{u} \preccurlyeq \mathbf{v}$ to denote $|u_j| < |v_j|$ and u_j, v_j have the same sign, for each index j. Obviously $\mathbf{u} \preccurlyeq \mathbf{v}$ indicates $\|\mathbf{u}\|_2 \leq \|\mathbf{v}\|_2$. In either case above, we have $|\mathcal{S}_t(w_j) - \mathcal{S}_t(w'_j)| \leq |w_j - w'_j|$, and sign preserves (or becomes zero), so $\mathcal{S}_t(\mathbf{w}) - \mathcal{S}_t(\mathbf{w}') \preccurlyeq \mathbf{w} - \mathbf{w}'$ for any threshold t. Therefore,

$$\begin{aligned}
\delta_{\mathbf{z}}^{k+1} =& \mathbf{z}^{k+1} - (\mathbf{z}')^{k+1} \\
=& \mathcal{S}_{\frac{\lambda}{\rho}}(\mathbf{x}^{k+1} + \mathbf{y}^k/\rho) - \mathcal{S}_{\frac{\lambda}{\rho}}((\mathbf{x}')^{k+1} + \mathbf{y}^k/\rho) \\
\preccurlyeq& \mathbf{x}^{k+1} + \mathbf{y}^k/\rho - ((\mathbf{x}')^{k+1} + \mathbf{y}^k/\rho) = \delta_{\mathbf{x}}^{k+1}
\end{aligned}$$

and

$$\begin{aligned}\delta_{\mathbf{y}}^{k+1} &= \mathbf{y}^{k+1} - (\mathbf{y}')^{k+1} \\ &= \mathbf{y}^k + \rho(\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) - \left(\mathbf{y}^k + \rho((\mathbf{x}')^{k+1} - (\mathbf{z}')^{k+1})\right) \\ &= \rho(\delta_{\mathbf{x}}^{k+1} - \delta_{\mathbf{z}}^{k+1}) \preccurlyeq \rho\delta_{\mathbf{x}}^{k+1}\end{aligned}$$

The last $\preccurlyeq$ holds because $\delta_{\mathbf{z}}^{k+1} \preccurlyeq \delta_{\mathbf{x}}^{k+1}$, the subtraction by $\delta_{\mathbf{z}}^{k+1}$ only pushes each element of $\delta_{\mathbf{x}}^{k+1}$ towards zero. So we have below conclusions for sensitivities of $\mathbf{x}, \mathbf{z}, \mathbf{y}$ after epoch k:

$$\Delta_2^{k+1}(\mathbf{x}) = \|\delta_{\mathbf{x}}^{k+1}\|_2 \leq \frac{2C}{n(1+\eta^{k+1}\rho)} \tag{4.13}$$

$$\Delta_2^{k+1}(\mathbf{z}) = \|\delta_{\mathbf{z}}^{k+1}\|_2 \leq \|\delta_{\mathbf{x}}^{k+1}\|_2 \leq \frac{2C}{n(1+\eta^{k+1}\rho)} \tag{4.14}$$

$$\Delta_2^{k+1}(\mathbf{y}) = \|\delta_{\mathbf{y}}^{k+1}\|_2 \leq \rho\|\delta_{\mathbf{x}}^{k+1}\|_2 \leq \frac{2\rho C}{n(1+\eta^{k+1}\rho)} \tag{4.15}$$

Theorem 4.3. *Algorithm 4.2 is (α, ϵ)-RDP.*

Proof. Let $\epsilon_{k+1,\mathbf{w}}(\alpha) = \alpha[\Delta_2^{k+1}(\mathbf{w})]^2/2\sigma^2$ for $\mathbf{w} \in \{\mathbf{x}, \mathbf{z}, \mathbf{y}\}$. By Lemma 2.1, according to the Gaussian mechanism, each epoch is $(\alpha, \sum_{\mathbf{w}\in\{\mathbf{x},\mathbf{z},\mathbf{y}\}} \epsilon_{k+1,\mathbf{w}}(\alpha))$-RDP, with respect to D. Since the algorithm has run T epochs, by Lemma 4, let $\epsilon = \sum_{k=1}^{T} \sum_{\mathbf{w}\in\{\mathbf{x},\mathbf{z},\mathbf{y}\}} \epsilon_{k,\mathbf{w}}(\alpha))$, then Algorithm 4.2 is (α, ϵ)-RDP. □

4.6 Experimental Results

In this section we will present our experimental results on both real and simulated datasets. We will first show performance of classification on two real datasets, then show performance of both classification and feature selection on a synthetic dataset.

4.6.1 ERM models

We perform our experiments on L_1 regularized logistic regression and huberized SVM. The objective function of logistic regression is given in (4.2). For huberized SVM, the objective function is

$$F(\mathbf{x}, D) := \frac{1}{n}\sum_{i=1}^{n} f_{\text{huber}}(l_i \mathbf{x}^T \mathbf{s}_i) + \lambda\|\mathbf{x}\|_1 \tag{4.16}$$

where

$$f_{\text{huber}}(w) := \begin{cases} 0 & \text{if } w > 1 + \hbar \\ \frac{1}{4\hbar}(1 + \hbar - w)^2 & \text{if } |1 - w| \leq \hbar \\ 1 - w & \text{otherwise} \end{cases}$$

is the huberized hinge loss (we set hyperparameter $\hbar = 0.5$ in all experiments).

4.6.2 Baselines

Many differentially private ERM algorithms cannot be applied to L_1 regularized classification, such as ObjPert [7, 12], OutPert [65], PVP and DVP [8], PSGD [9], and RSGD [10]. Therefore, we compare our proposed algorithms with these baselines: DP-SGD [14], DP-ADMM [55], ADMM-objP [52], and Non-Private approach.

DP-SGD performs stochastic gradient descent with Gaussian perturbation. To handle the L_1 regularization, when the algorithm requires taking gradient on $f(\mathbf{x}^k, B_k) + \lambda\|\mathbf{x}^k\|_1$, we use the proximal gradient technique

$$\mathbf{x}^{k+1} \leftarrow \mathcal{S}_{\lambda\eta^k}[\mathbf{x}^k - \eta^k \nabla f(\mathbf{x}^k, B_k)] \tag{4.17}$$

to update $\mathbf{x}^{k+1}$, as suggested in [66] and [67]. DP-ADMM is a distributed learning version of ADMM, where each party transfers perturbed primal variables to the center, and the center draws a consensus of the parties, then transfers primal and dual variables back to each party. ADMM-objP is an ADMM version of the objective perturbation algorithm. At each iteration, the trainer optimize a perturbed non-regulated objective function, therefore although the algorithm satisfies ϵ-DP, in practice it is not really differentially private due to the objective function can only be approximately solved. According to their paper, we apply gradient descent enough times and assume the optimization problem is exactly solved at each iteration.

The DP-SVRG algorithm presented in [15] can also be applied on non-smooth regularizers, but we have implemented and found that, due to the extra privacy budget required to spent on perturbing the full gradient, with the high privacy range ($\epsilon \leq 1$), if we choose a large noise scale σ^2, the perturbed full gradient cannot help as a control variant to fasten the training, but actually slows down the minimization of empirical loss; if we choose a small σ^2, the privacy budget accumulates too fast and exceed our range in a few iterations. Therefore we have dropped this algorithm in our comparisons.

4.6.3 Datasets and Pre-proessing

Two real datasets on human subjects were used in our study: (i) the Adult dataset [31] was generated from 1994 US Census, with $n = 48,842$, $p = 124$, and the frequency of the majority label is 0.761; (ii) the IPUMS-BR dataset [33] was extracted from IPUMS data, with $n = 38,000$, $p = 53$, and the frequency of the majority label is 0.507.

An intercept is added into each dataset. All numerical attributes are re-scaled into [0, 1] by Min-Max scalar. For the algorithms requiring feature vector to have bounded L_2 norm, we normalize to make $\|\mathbf{x}_i\|_2 \leq 1$ for $i = 1, ..., n$.

To test the performance on feature selection, we created a synthetic sparse dataset with many irrelevant features, using similar strategy as in [52]. To be specific, we generate a 100-dimension data $\mathbf{s}_i \sim \mathcal{N}(0_{100}, \Sigma)$ for $i = 1, ..., n$, where the variance-covariance matrix $\Sigma_{j,k} = 0.5^{|j-k|}$ for $j, k = 1, ..., 100$. Let $\mathbf{m}$ be the true model, defined as $\mathbf{m}_{1:10} = (0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5)$, $\mathbf{m}_{11:20} = -\mathbf{m}_{1:10}$, and $\mathbf{m}_{21:100} = (0, ..., 0)$. For the label of each row l_i, we sample the Bernoulli distribution with $\Pr(l_i = 1) = 1/(1 + \exp(-\mathbf{m}^T\mathbf{s}_i + \iota))$, where $\iota \sim \mathcal{N}(0, 1)$ is a random noise. Therefore, to predict l_i, $\mathbf{s}_i$ contains 20 relevant features and 80 irrelevant features. We generate $n = 40,000$ samples to constitute one dataset, the frequency of the majority label is 0.500. We perform L_1 regularized logistic regression on simulated data, since it is usually used for attribute selection.

We did 10-fold cross validation on each experiment for each algorithm, and due to randomness from noisy injection, we repeat each fold 10 times and report average classification accuracy and objective value on testing data. For the simulated data, we generated 10 datasets using the simulation strategy, and report the average performance.

4.6.4 Hyperparameter setting

We keep $\delta = 10^{-8}$ for all experiments. For those algorithms satisfying RDP, we choose the best conversion to (ϵ, δ)-DP. In non-private settings, model users usually train a series models with different candidates of regularization coefficient λ, and select the one with highest testing performance. However, this process is data-dependent, therefore in private settings we cannot take a "best performing" coefficient for granted. Instead, we performed two group of experiments by two frequently chosen coefficients in L_1 regularized classification: low regularization with $\lambda = 0.0001$ and high regularization with $\lambda = 0.001$.

For ssADMM and DP-SGD, we set mini-batch size $m = \sqrt{n}$. We choose $\eta^k = \eta^0/h$ where h is the current expected epoch (we consider every n/m iterations as one expected epoch), since we find this schedule has the best performance for both algorithms, compare to a constant learning rate, or a decreasing one at a rate of $O(1/\sqrt{h})$. After tuning on the simulated data, we set penalty term $\rho = 0.25$ for ssADMM and $\rho = 0.5$ for mpADMM. For mpADMM, we use a constant learning rate. For DP-ADMM, we assume there are 2 parties, each holding half of the data. (If there is only one party, DP-ADMM will reduce to DP-SGD with sampling ratio $q = 1$.) For ADMM-objP, at each iteration we optimize the perturbed objective function by full gradient descent running 20 epochs. Other hyperparameters for DP-ADMM and ADMM-objP are set according to their papers.

4.6.5 Classification Performance on Real Data

Figure 4.1 and Figure 4.2 plots the testing data accuracy (top) and objective values (bottom) of the algorithms trading off with privacy parameter ϵ, for L_1 regularized logistic regression and huberized SVM, respectively. We can see that for classification accuracy, ssADMM outperforms other algorithms in most cases. Although ssADMM only performs slightly better than DP-SGD in IPUMS-BR data in classification accuracy, the advantage of ssADMM is more obvious in its objective value. This is in accordance with the experiment in [42] that sADMM outperforms proximal gradient in non-private setting. [68] also show that ADMM based algorithms are more robust to noisy data with outliers. Comparing mpADMM with DP-SGD, although DP-SGD has better classification accuracy than mpADMM in some cases, its objective value is usually outperformed by mpADMM. mpADMM performs better in adult dataset than in IPUMS-BR dataset, probably because Adult dataset is more sparse compare to IPUMS-BR, due to it is binary transferred through one-hot encoding. This robustness of model perturbation to data with irrelevant attributes is in accordance with our observations on the simulated data. DP-ADMM and ADMM-objP can achieve high utility when ϵ gets high, but in our testing range of ϵ, they cannot perform as good as other algorithms.

4.6.6 Performance on Simulated Data

To measure the attribute selection performance, we test how many relevant attributes are selected by each algorithm for L_1 regularized logistic regression. Since the dataset is standardized, we can use the magnitude

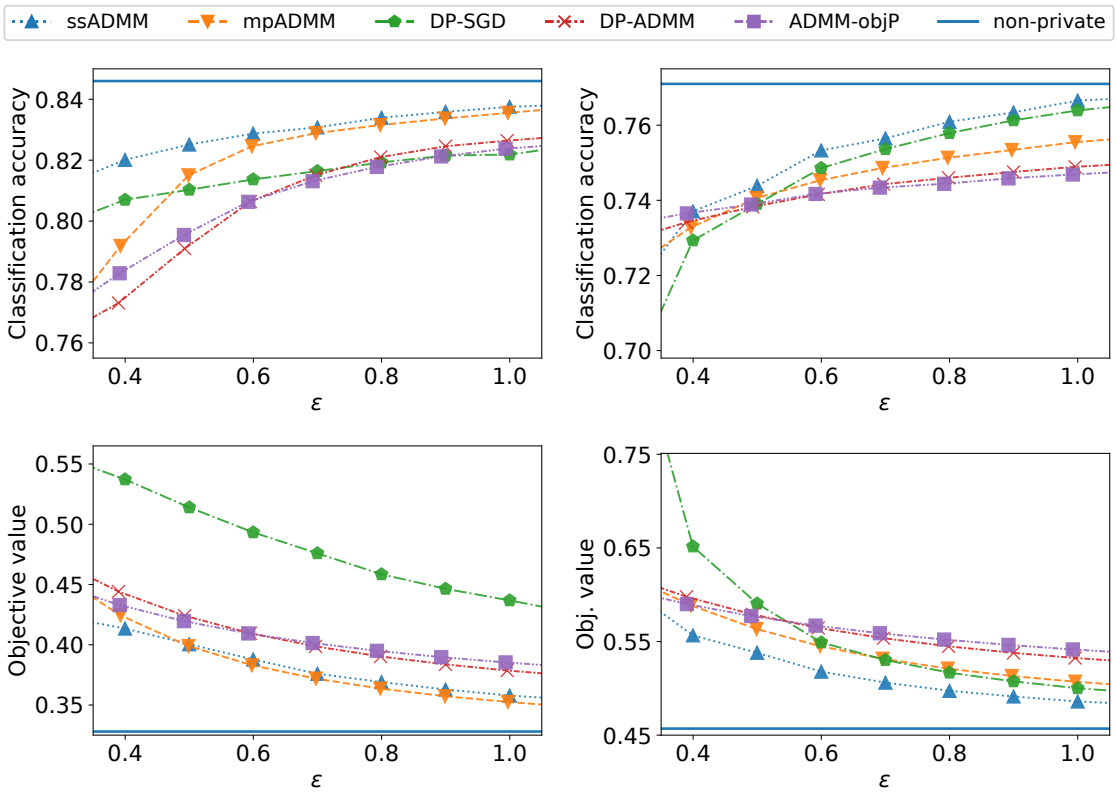
of the coefficient to rank the attributes, due to that noisy perturbation might cause the coefficients of irrelevant attributes slightly differ from zero.

We define a criterion ξ_K to measure the coverage of true relevant attributes if top K attributes suggested by the attribute ranker were selected. For example, since we know there are 20 relevant attributes in the simulated data, if we select $K = 30$ attributes by magnitude of coefficient, 16 of them are the true relevant ones (i.e. among 1 to 20 in our synthetic dataset), then $\xi_{30} = 16/20 = 0.8$. This criterion is reasonable because in real scenario, an attribute ranker usually pre-determines the number of attributes it shall select, and select the top K ranked attributes, where K depends on the number of attributes the user can afford. We test all algorithms for $K = 20, 25, 30,$ and 40.

Figure 4.3 shows the classification performance of each algorithm on the simulated data. For non-private performance, we assume the true model is known. We can see that SSADMM, MPADMM, and DP-SGD have similar performance in classification accuracy. Figure 4.4 shows the performance of attribute selection. Although classification accuracy are close, we can see that MPADMM can detect more relevant attributes, especially in the low ϵ range. ADMM-OBJP, an algorithm which was originally proposed for feature selection, can outperform SSADMM and DP-SGD for feature selection in low ϵ while its classification accuracy is behind SSADMM and DP-SGD. However, ADMM-OBJP usually require much more epochs in training compare to the other algorithms. Therefore, if we know the data is sparse and the major goal is focused on attribute selection, MPADMM is more preferable.

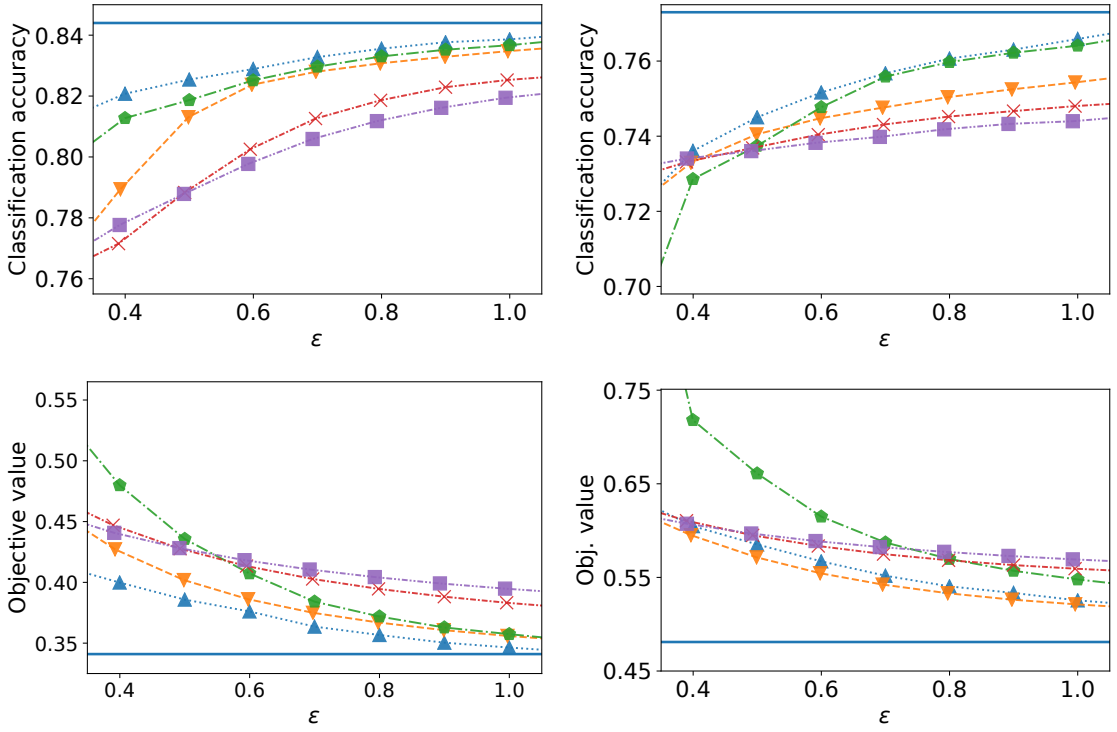
4.7 Conclusions

We present two privatizations of stochastic ADMM under Rényi differential privacy. One algorithm combines gradient perturbation technique with privacy amplification result to reduce the total privacy loss throughout the execution. The other algorithm uses the output perturbation (with numerical computation of sensitivity) to privately release the solution at the end of each training epoch. These algorithms can be used to solve optimization problems with complex structural regularizations that induce sparsity.



(a) Adult $\lambda = 0.0001$

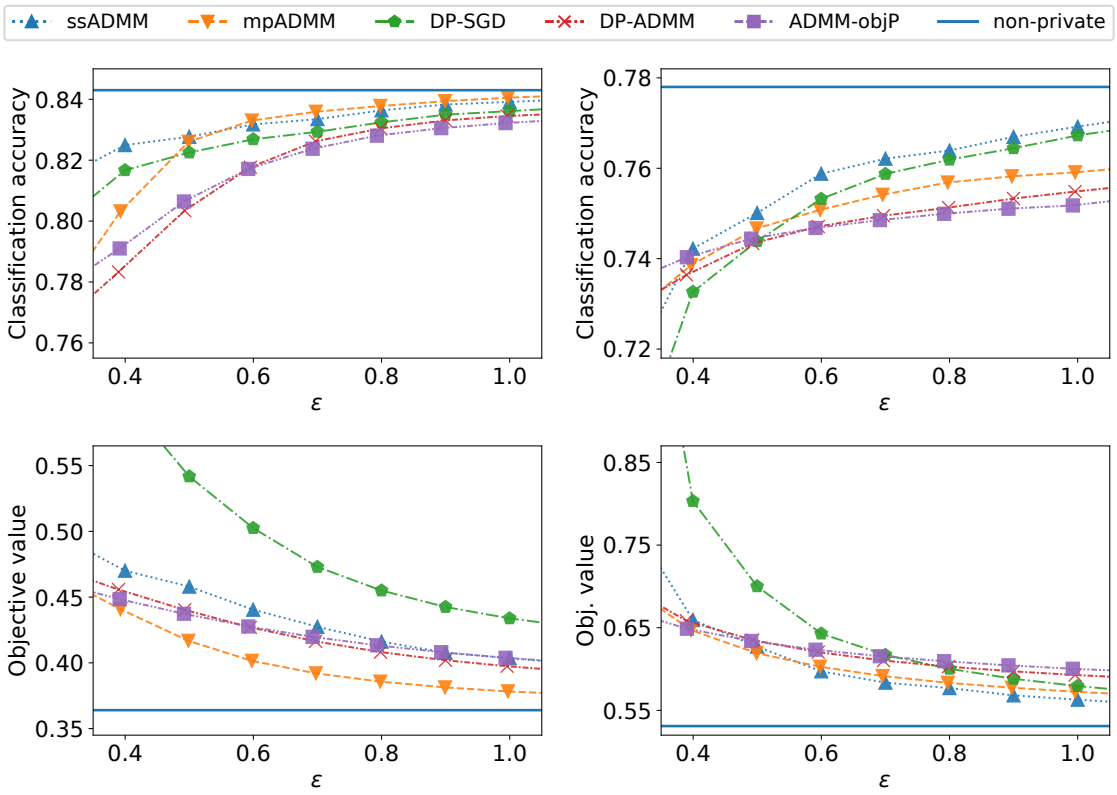
(c) IPUMS-BR $\lambda = 0.0001$



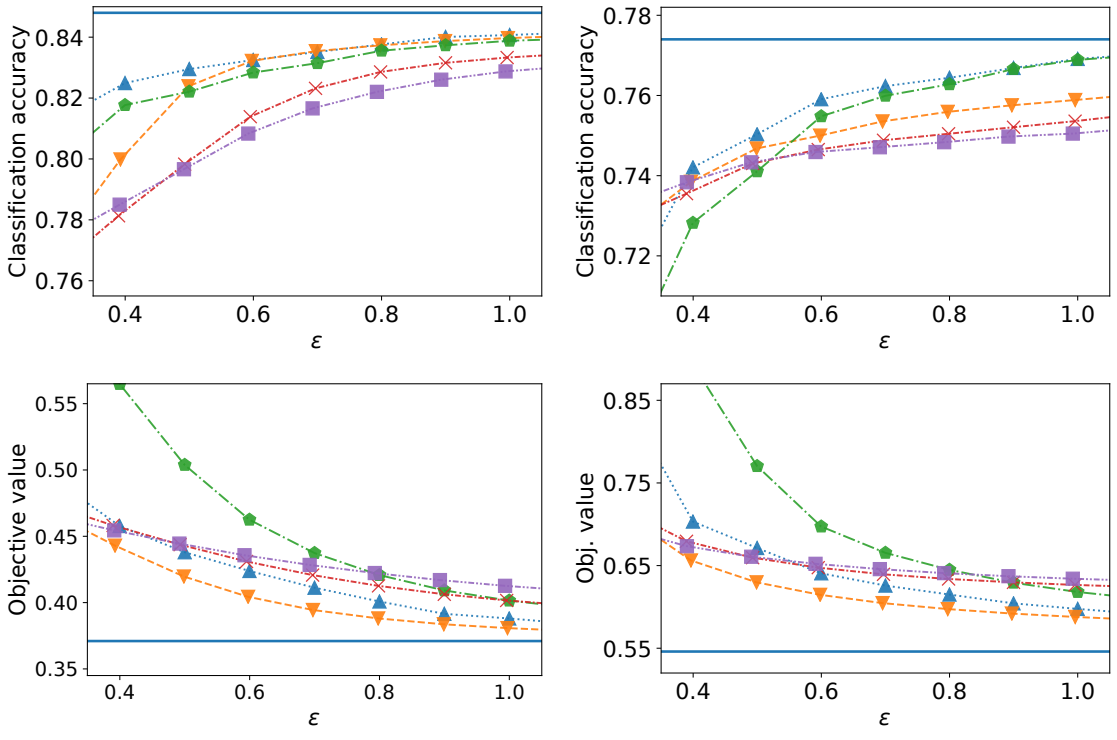
(b) Adult $\lambda = 0.001$

(d) IPUMS-BR $\lambda = 0.001$

Figure 4.1: Logistic regression result by ϵ (Top: Classification accuracy; Bottom: Objective value)

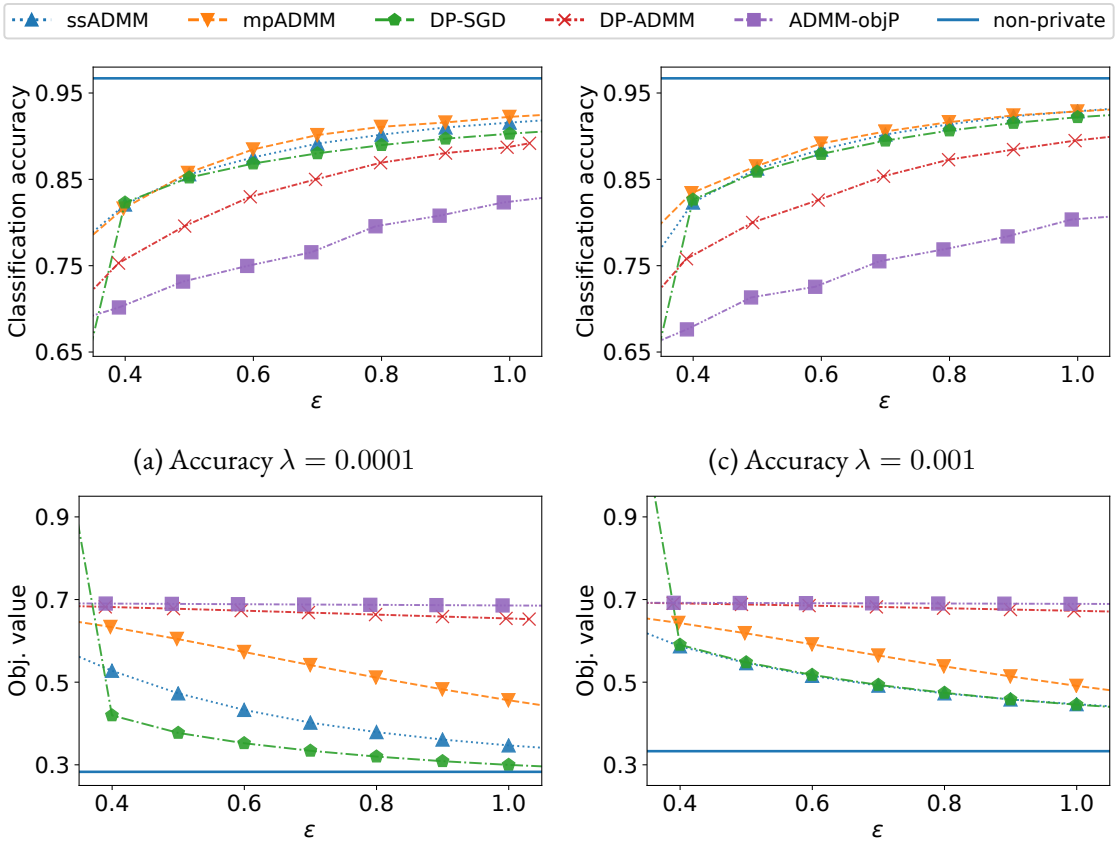


(a) Adult $\lambda = 0.0001$ (c) IPUMS-BR $\lambda = 0.0001$



(b) Adult $\lambda = 0.001$ (d) IPUMS-BR $\lambda = 0.001$

Figure 4.2: Huberized SVM result by ϵ (Top: Classification accuracy; Bottom: Objective value)



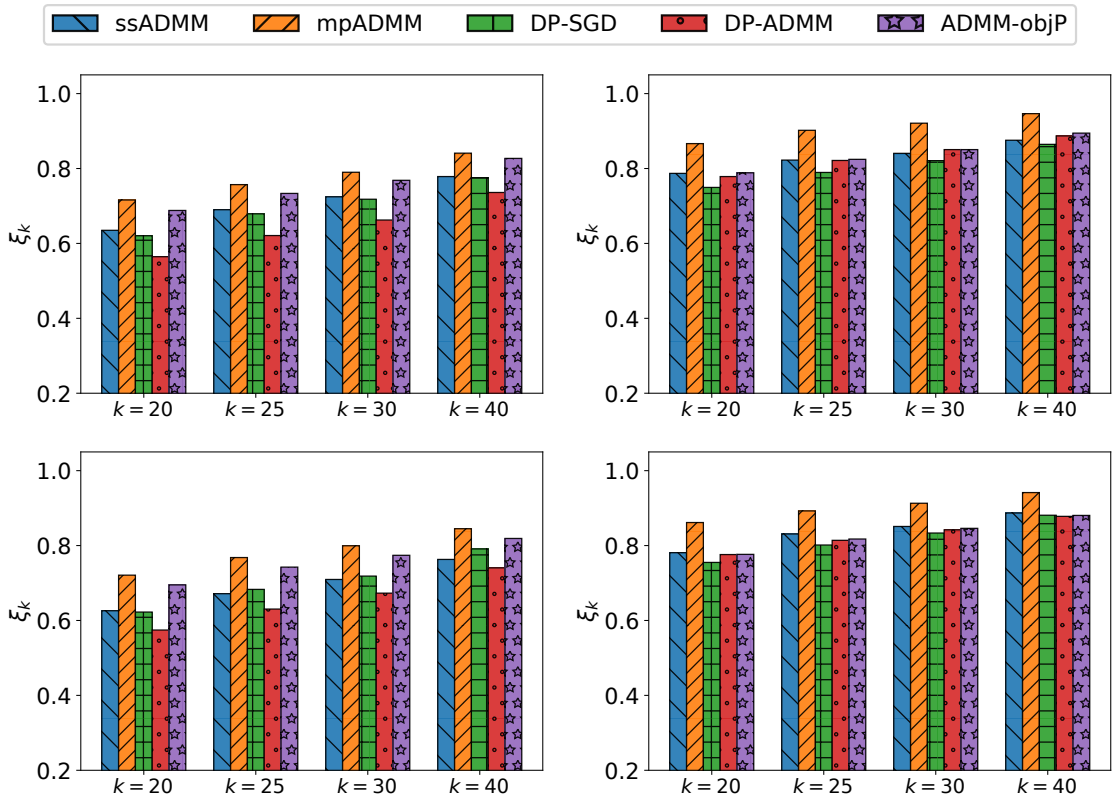
(a) Accuracy $\lambda = 0.0001$

(c) Accuracy $\lambda = 0.001$

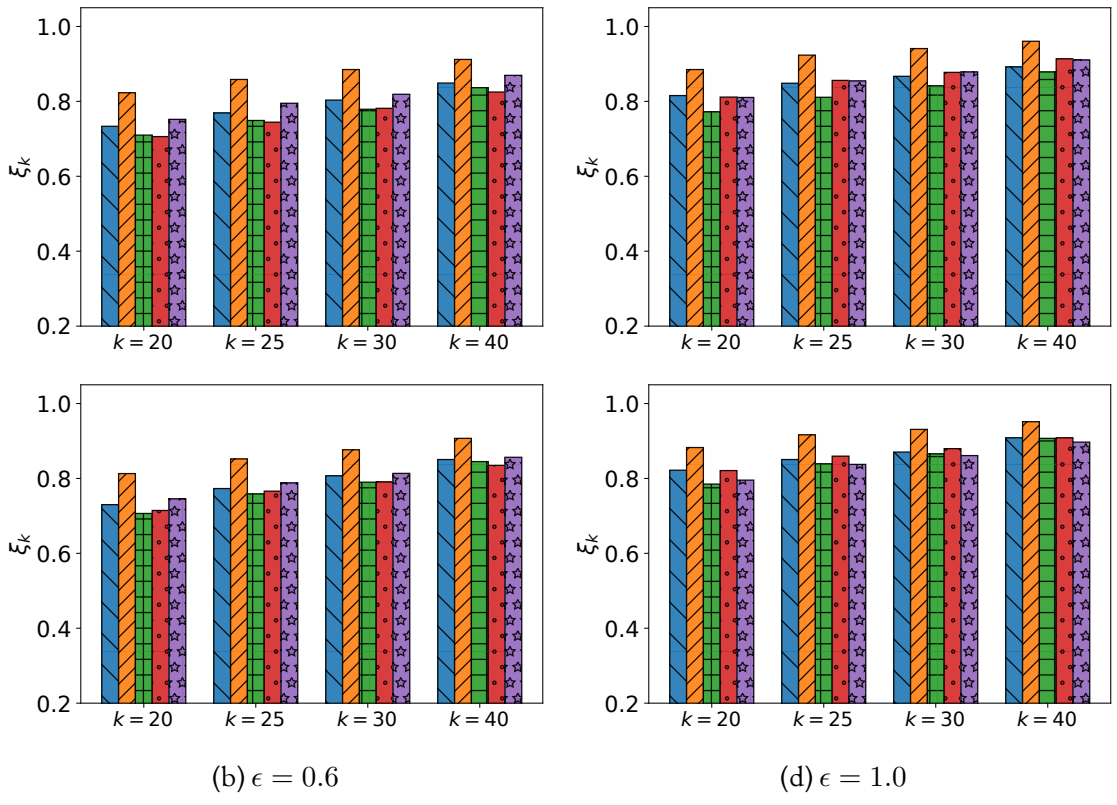
(b) Objective value $\lambda = 0.0001$

(d) Objective value $\lambda = 0.001$

Figure 4.3: Classification performance on simulated data



(a) $\epsilon = 0.4$ (c) $\epsilon = 0.8$



(b) $\epsilon = 0.6$ (d) $\epsilon = 1.0$

Figure 4.4: Attribute selection performance on simulated data (Top: $\lambda = 0.0001$; Bottom: $\lambda = 0.001$)

Chapter 5

Stochastic Adaptive Line Search for Differentially Private Optimization

5.1 Motivation

Gradient perturbation methods can be applied on a broader range of optimization problems, including the non-convex ones, like neural network. Therefore, it is of great importance to improve the performance of gradient perturbation algorithms to achieve better utility. One key problem of private stochastic gradient descent is that, the stochastic gradients would not point to a descent direction to optimize the objective function after noisy perturbation, and there is no a apriori way to select an appropriate step size without extra privacy cost. In this chapter, an adaptive step size selection method is proposed to improve the optimization efficiency, and the privacy leak of the objective evaluation is handled by the Sparse Vector Technique. Previous privacy framework used on gradient perturbations, such as strong composition [13] and moment accountant [14], can only accumulate privacy leak from Gaussian mechanism, therefore it is difficult to tightly composite an extra source of privacy leak. However, the recent Rényi differential privacy framework provides a descent composition for multiple RDP mechanisms, which makes it possible to integrate adaptive step size selection into private optimization. [1]

[1] This chapter is a slightly modified version of [69] published in IEEE Big Data 2020 Proceedings and has been reproduced here, complying with author rights.

5.2 Introduction

We consider solving the following finite-sum optimization problem under differential privacy [2,4,70–72]:

$$\arg\min_{\mathbf{w}\in\Theta} F(\mathbf{w}; D) := \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{w}; \mathbf{d}_i)\,, \tag{5.1}$$

where $D = \{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$ is i.i.d. examples drawn from an unknown data distribution and f represents the loss on one training example. This formulation includes a wide range of machine learning problems, for example, training a neural network with weights $\mathbf{w}$ for classification. Stochastic gradient descent (SGD) has been widely used, especially for large-scale problems, to solve the problem of form (5.1) due to its simplicity and low iteration cost. For differential privacy, the SGD update typically has the form of:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\mathbf{g}_t + Y(\epsilon_t)\right),$$

where $\eta_t > 0$ is a step size, Y is a noise (e.g., Gaussian) variable whose scale is determined by the per-iteration privacy budget ϵ_t, and $\mathbf{g}_t$ is the gradient evaluated on a subset $B_t \subseteq D$ of examples selected for iteration t:

$$\mathbf{g}_t = \frac{1}{|B_t|}\sum_{d_i\in B_t} \nabla f(\mathbf{w}_t; \mathbf{d}_i).$$

Despite its prevalent use in differentially private optimization, the use of SGD in practice faces two major challenges. First, the direction pointed by the stochastic gradient $\mathbf{g}_t$ may not be a descent direction. Even worse, depending on the magnitude of noise $Y(\epsilon_t)$, the update direction may still not be a descent direction even when $\mathbf{g}_t$ is one. A natural question is how to decide whether the privacy budget ϵ_t is sufficiently large enough to get the learning signal, i.e., $\mathbf{g}_t$ is not dominated by $Y(\epsilon_t)$. Second, the efficiency of SGD largely relies on the choice of step size η_t. It can be chosen independent of data, e.g., a constant step size [13, 14]. However, these step sizes are often problem-specific and require a degree of fine-tuning. The methods with data-dependent step sizes [17] require allocating extra privacy budget for selection and efficiently controlling the growth rate of cumulative budget.

In this work, we propose a Rényi differentially private backtracking line search algorithm that adaptively sets the step size using the Armijo condition and empirically show that it can improve the performance of algorithm on both convex and non-convex problems. Armijo line search [24, 73] is a classical

technique to find a step size η that gives sufficient reduction in the objective function f. Recently, [74] introduced a stochastic version in which both objectives and gradients are approximated using a random subset of data. To be specific, it uses backtracking algorithm to find a step size η that satisfies

$$f_B(\mathbf{w}_t - \eta \nabla f_B(\mathbf{w}_t)) \leq f_B(\mathbf{w}_t) - \alpha\eta \|\nabla f_B(\mathbf{w}_t)\|_2^2 \,, \tag{5.2}$$

where $\alpha \in (0, 1)$ is a hyperparameter and $f_B(\cdot)$ denotes that f is evaluated on the minibatch B. However, privatizing the Armijo line search is a non-trivial task. A naive privatization of this search algorithm may require unacceptably large privacy budget as it requires multiple function evaluations on the dataset. Motivated by the observation that the Armijo line search sequentially evaluates *threshold* queries

$$q(\eta) = f(\mathbf{w}_t) - f(\mathbf{w}_t - \eta \nabla f(\mathbf{w}_t)) - \alpha\eta \|\nabla f(\mathbf{w}_t)\|_2^2 \geq 0$$

for different values of η, the proposed algorithm adopts the Sparse Vector technique [70, 75], which allows the algorithm to pay the privacy budget only for η that satisfies the condition. Applying the sparse vector algorithm on a randomly subsampled data further allows the algorithm to relax the budget constraint using the recent privacy amplification results [6]. While in a deterministic (i.e., noise-free) setting, it is guaranteed that there exists η that satisfies the condition (5.2), in a stochastic private setting, the backtracking algorithm may fail to terminate or return an arbitrarily small step size due to the noise from two different sources: (i) gradient approximation and (ii) noise added for privacy. When the backtracking algorithm fails to return within the pre-specified number of iterations, to decide whether more accurate gradients are necessary, the proposed algorithm evaluates another gradient at $\mathbf{w}_t$ and measure the angle between two gradients. When two gradients evaluated at $\mathbf{w}_t$ are pointing to very different directions, the algorithm increases the privacy budget for gradient evaluation.

Our contributions are summarized as follows:

- We propose a Rényi differentially private SGD with Armijo line search. To the best of our knowledge, this is the first private SGD algorithm with line search ability.

- We introduce an adaptive privacy budget controlling strategy based on the moving average of angles between consecutive gradients, which detects if gradients are pointing to very different directions.

- To evaluate the effectiveness of the proposed algorithm, we conduct extensive experiments on real datasets and compare its performance to existing algorithms.

The rest of this chapter are organized as: Section 5.3 reviews the related work; Section 5.4 summarizes important definitions and lemmas used in this chapter; Section 5.5 presents the main algorithm; and experimental results are presented in Section 5.6; Section 5.7 concludes the chapter.

5.3 Related Work

Many techniques have been proposed for first-order optimization algorithms in non-private setting, focusing on step size selection or reducing the noise involved in stochastic gradients, such as Adam [76], SVRG [21], SplitSGD [77], etc. The technique related to this chapter is Amijo line search [73], which is a classic and famous step size selection method. Recent works [74, 78] have shown that combining SGD with line-search achieves fast convergence for both convex and non-convex problems, and is robust to the precise choices of hyperparameters, for over-parameterized models, with the price of additional objective evaluation (feed-forward steps for neural networks). In this chapter, we show that, with essential randomization techniques, it can fit well into the privacy framework, and the privacy budget can be carefully controlled.

There are many differentially private mechanisms we can use to release various statistics. One advanced tool highly related to this chapter is the sparse vector technique (SVT) [70]. The sparse vector algorithm sequentially processes a sequence of threshold queries. For each query in the sequence, the algorithm evaluates it with noise, compares the result with the noisy threshold, and outputs the binary value. The carefully scaled noise ensures that the algorithm only pays the privacy budget when the query is privately evaluated as above the threshold. Although [75] shows that many extensions of SVT are not private, it also demonstrates the correctness of the original version (used in our approach), which is further confirmed in [79].

Differentially private optimization algorithms can be roughly grouped into three categories. Output perturbation algorithms train a model without noisy perturbation, then perturb the model before releasing, based on a calculated sensitivity, such as [7–10]. Objective perturbation algorithms protect the privacy of the training data through optimizing a noise-perturbed objective, for example, [7, 12]. The aforementioned algorithms usually put strict assumptions on the objective functions, such as convexity

and smoothness, which limits their applicable domain. The type of algorithms mostly related to this chapter is the gradient perturbation algorithms, which perturb the data-dependent intermediate results (i.e. gradients) during the model training, and the total privacy is calculated by composing the privacy costs of all iterations. Since privacy is achieved immediately after the data-dependent step, gradient perturbation algorithms do not put strict assumptions on the objective function, and can be applied in a broader range of problems, such as neural networks. The first gradient perturbation algorithm was proposed in [13], with the "strong composition" method to account for the privacy loss over multiple iterations. Later, "moment accountant" method [14] gave a tighter bound on privacy amplification and accountant. These algorithms directly satisfy (ϵ, δ)-differential privacy, and many algorithms were built based on them, such as [15, 45]. These algorithms take gradient calculation as the only data-dependent step, thus there is no extra source of information which can be used to adaptively tune hyperparameters such as step size, per-iteration privacy budget, and/or clipping threshold. [17] is an exception, which proposed an adaptive gradient perturbation algorithm based on full gradient descent, with extra budget paid for objective evaluation, and it satisfies zCDP, without privacy amplification. The convergence property of private optimization algorithms is shown in [80, 81].

Rényi differential privacy (RDP) is a recent privacy framework proposed in [4], which stands between pure and approximate DP, and its privacy amplification lemma is presented in [6]. The privacy guarantee of our algorithm fits into the RDP framework, and we show that it can help account for the two sources of privacy leaks. This differs from the "moment accountant" technique, which is tightly coupled with the Gaussian mechanism, and also different from the zCDP framework, which does not yet have privacy amplification due to sub-sampling.

5.4 Preliminaries

Two datasets D and D' are considered to be *neighboring* if they differ by one individual, i.e., $|(D \setminus D') \cup (D' \setminus D)| = 1$, denoted by $D \sim D'$. We use bold-face letters to represent vectors and a subscript to indicate iteration number (e.g. $\mathbf{w}_t$ denotes the value of $\mathbf{w}$ at iteration t). A summary of symbol notations in this chapter is presented in Table 5.1.

Table 5.1: Summary of symbol definitions

Symbol	Definition	Symbol	Definition
D	dataset	α	order of Rényi divergence and RDP
$\mathbf{d}$	a datum	ϵ	privacy budget of DP or RDP
n	data size	δ	privacy parameter of approximate DP
F	loss function on dataset	γ	Gaussian noise sample
f	loss function on one datum	σ^2	variance of Gaussian noise
i	iterate through a dataset	λ, ν	Laplace noise sample
t	iteration number of model training	ρ	privacy budget independent of α
q	sampling ratio	η	step size (learning rate)
B	subsampled mini-catch	Ω	set of selected step sizes
$\mathbf{w}$	model vector	ξ	budget adaptation factor
Θ	model space	θ	angle between gradients
$\mathbf{x}$	feature vector of one datum	ψ	decaying factor
y	label of one datum	ϕ	threshold of angle
$\mathbf{g}$	gradient vector	τ	iterations to reset selected learning rates
C	clipping threshold	ζ	clipping threshold adaptation factor
α, β	Armijo line search hyperparameter	μ	regularization coefficient

5.4.1 Sparse Vector Technique

The Sparse Vector is a technique used to answer a sequence of threshold queries $\{q_i\}, i = 1, 2, \cdots$. Given a publicly known threshold T, it sequentially processes each q_i and produces an output $a_i \in \{\top, \bot\}$. Each a_i indicates whether $q_i(D)$ is above ($\top$) or below ($\bot$) the threshold. It terminates after outputting the predefined number c of "$\top$" values, and its privacy cost is proportional to c. In other words, given a fixed privacy budget, it can release binary answers to threshold queries until it outputs c "above" threshold answers regardless of how many "below" threshold answers are generated. AboveThreshold is a basic version with $c = 1$.

Lemma 5.1 (Above Threshold Mechanism)**.** *[70] Let $\{q_i\} = q_1, q_2, ...$ be a series of queries having the same L_1 sensitivity $\Delta_1(q)$, and T be a publicly known threshold. The AboveThreshold algorithm first perturbs T by adding Laplace noise, i.e, $\hat{T} = T + \lambda$ where $\lambda \sim Lap(0, \frac{2\Delta_1(q)}{\epsilon})$ and generates output*

$\{a_i\}$ *as follows.*

$$a_i = \begin{cases} \top & \text{if } q_i(D) + \nu_i \geq \hat{T}, \\ \bot & \text{if } q_i(D) + \nu_i < \hat{T}, \end{cases}$$

where $\nu_i \sim Lap(0, \frac{4\Delta_1(q)}{\epsilon})$. *The mechanism terminates if* $a_i = \top$. *The* AboveThreshold *satisfies* $(\epsilon, 0)$*-DP.*

5.4.2 Rényi Differential Privacy

The definition and basic properties of Rényi differential privacy (RDP) can be found in Chapter 2. In addition, to quantify the per-iteration privacy budget for gradient evaluation, we can define a "privacy budget" independent of α, used to determine the scale of noise, σ^2. According to Lemma 2.1, the Gaussian mechanism with noise scale σ^2 and L_2 sensitivity $\Delta_2(q)$ ensures $\mathcal{M}$ to satisfy $(\alpha, \alpha\rho)$-RDP for $\alpha > 1$, [2] where

$$\rho := \Delta_2^2(q)/(2\sigma^2)$$

can be considered as such a "privacy budget". Multiply ρ by α would give the privacy budget ϵ spent under order α.

Another lemma which can be used to connect pure DP and RDP is presented below.

Lemma 5.2 (ϵ-DP to RDP)**.** *[3] If* $\mathcal{M}$ *satisfies* $(\epsilon, 0)$*-DP, then the Rényi divergence*

$$\mathbb{D}_\alpha[\mathcal{M}(D) \| \mathcal{M}(D')] \leq \frac{1}{2}\alpha\epsilon^2 \,.$$

In other words, $\mathcal{M}$ *also satisfies* $(\alpha, \frac{1}{2}\alpha\epsilon^2)$*-RDP.*

This lemma holds regardless of which mechanism $\mathcal{M}$ uses to achieve differential privacy.

[2] To avoid confusion, in this chapter, we use the curly α to denote the order of Rényi divergence and RDP, and plain α to denote the hyperparameter in Armijo condition.

5.5 Algorithms

This section describes each component of the proposed algorithm in detail. Starting with a private backtracking line search algorithm to find η for a given gradient, followed by the main algorithm, and some heuristic improvements.

5.5.1 Noisy Backtracking Line Search

We start with Noisy Backtracking Line Search (NoisyBTLS) algorithm which performs backtracking line search in a differentially private manner. The pseudocode of the algorithm is shown in Algorithm 5.1. NoisyBTLS is an application of AboveThreshold algorithm [70], introduced in Lemma 5.1, to a line search task.

The algorithm starts by adding noise to the threshold $T = 0$, producing a noisy threshold $\hat{T} = \lambda$, where λ is a random noise drawn from a Laplace distribution. Instead of Laplace noise, one can also chose to add Gaussian noise in Algorithm 5.1. We show in Theorem 5.3 that the algorithm with Gaussian noise satisfies RDP. At each iteration, the algorithm evaluates a query $q_i(\eta, D) = f(\mathbf{w}) - f(\mathbf{w} - \eta\nabla f(\mathbf{w})) - \alpha\eta\|\nabla f(\mathbf{w})\|_2^2$ with noise ν_i and compares it (i.e., $q_i(\eta, D) + \nu_i$) with the noisy threshold $\hat{T}$. If $q_i(\eta, D) + \nu_i \geq \hat{T}$, the algorithm outputs η and halts. Otherwise, it decreases the step size η by multiplying with β and continues with the next iteration. Here, $\beta \in (0, 1)$ is a user-defined multiplicative factor that determines how fast the step size is decreased. One crucial difference with the original Armijo line search algorithm is that we set a limit on the number of iterations. If there is no limit, when the query value is dominated by noise, it would fail to terminate or returns a too small step size, which does not help make progress and could lead to increase in the objective value at the next iteration. Hence, when the algorithm fails to return within the specified maximum number of iterations, the algorithm computes a diagnostic statistic to test whether higher privacy budget is necessary and adjusts the budget according to the test result. We discuss details of this procedure in Section 5.5.2. The use of Sparse Vector technique in Algorithm 5.1 significantly reduces the privacy budget needed to find η, from a scale linear to the size of the search space to a constant, which greatly improves its utility. A naive implementation would result in $(\epsilon_1 + \texttt{max\_it} \cdot \epsilon_2, 0)$-DP.

Algorithm 5.1 Noisy Backtracking Line Search (NoisyBTLS), Laplace [resp. Gaussian] version

Input: Objective function f, dataset D, model parameter $\mathbf{w}$, gradient $\mathbf{g}$, initial learning rate η_0, privacy budget ϵ_{BT} [resp. ρ_{BT}], sensitivity Δ_q.
Hyper-parameters: α, β, maximum iterations `max_it`.
$\epsilon_1 \leftarrow \frac{\epsilon_{BT}}{2}, \epsilon_2 \leftarrow \frac{\epsilon_{BT}}{4}$ [resp. $\sigma_1^2 \leftarrow \frac{3}{2\rho}, \sigma_2^2 \leftarrow \frac{3}{\rho}$]
Sample noisy threshold $\hat{T} = \lambda$, where $\lambda \sim Lap(0, \frac{\Delta_q}{\epsilon_1})$ [resp. $\lambda \sim \mathcal{N}(0, \Delta_q^2\sigma_1^2)$]
$\eta \leftarrow \eta_0$
for $i = 1, 2, \ldots,$ `max_it` **do**
    $q_i \leftarrow f(\mathbf{w}; D) - \alpha\eta\|\mathbf{g}\|_2^2 - f(\mathbf{w} - \eta\mathbf{g}; D)$
    $\hat{q}_i \leftarrow q_i + \nu_i$ where $\nu_i \sim Lap(0, \frac{\Delta_q}{\epsilon_2})$ [resp. $\nu_i \sim \mathcal{N}(0, \Delta_q^2\sigma_2^2)$]
    if $\hat{q}_i \geq \hat{T}$ **then**
        Output: η ▷ found a suitable step size
    end if
    $\eta \leftarrow \beta\eta$
end for
Output: 0 ▷ failed to find η within `max_it` iterations

Theorem 5.1. *Let Δ_f be an upper bound on the objective function f such that $|f(\mathbf{w}; \mathbf{d})| \leq \Delta_f$ for $\forall \mathbf{d} \in \mathcal{D}$ and $\mathbf{w} \in \Theta$. Given the candidate gradient $\mathbf{g}$ either privately released or publicly available, Algorithm 5.1 with Laplace noise, $\epsilon_1 = \frac{\epsilon}{2}, \epsilon_2 = \frac{\epsilon}{4}$, and $\Delta_q = \Delta_f$ satisfies $(\epsilon, 0)$-DP.*

Proof. Consider the query q_i, q_i' evaluated on datasets D and D', respectively, where D and D' differs by the presence or absence of one datum d_*:

$$
\begin{aligned}
&|q_i - q_i'| \\
=&|[f(\mathbf{w}, D) - \alpha\eta\|\mathbf{g}\|^2 - f(\mathbf{w} - \eta\mathbf{g}, D)] - [f(\mathbf{w}, D') - \alpha\eta\|\mathbf{g}\|^2 - f(\mathbf{w} - \eta\mathbf{g}, D')]| \\
=&|f(\mathbf{w}, D) - f(\mathbf{w}, D') - [f(\mathbf{w} - \eta\mathbf{g}, D) - f(\mathbf{w} - \eta\mathbf{g}, D')]| \\
=&|f(\mathbf{w}, d_*) - f(\mathbf{w} - \eta\mathbf{g}, d_*)| \leq \Delta_f
\end{aligned}
$$

The last equality holds regardless of whether $D \backslash D' = \{d_*\}$ or $D' \backslash D = \{d_*\}$. The last inequality holds since both $f(\mathbf{w}, d)$ and $f(\mathbf{w} - \eta\mathbf{g}, d)$ are non-negative within range $[0, \Delta_f]$.

Therefore, Algorithm 5.1 is applying the AboveThreshold mechanism (Lemma 5.1), with $q_1, q_2, \ldots$, each has sensitivity Δ_f, comparing $f(\mathbf{w}, D) - \alpha\eta_i\|\mathbf{g}\|^2 - f(\mathbf{w} - \eta_i\mathbf{g}, D)$ with public thresh-

old $T = 0$. (Assume there is a dummy query after q_{max_it} which always return true.) So, according to Lemma 5.1, Algorithm 5.1 is ϵ-DP. □

Theorem 5.1 requires f is upper bounded by a constant Δ_f. If there is no a priori known upper bound on a loss function f, we enforce the bound by applying the objective clipping [17]: $f(\mathbf{w}; D) = \sum_{i=1}^{n} \min\{f(\mathbf{w}; \mathbf{d}_i),\ \Delta_f\}$. Since the Laplace version of Algorithm 5.1 is ϵ-DP, one can use Lemma 5.2 to convert its privacy guarantee to that of RDP. Instead, in the following theorem, we directly derive the Rényi divergence of output distributions between two neighboring datasets and show it results in a tighter bound on the privacy loss.

Theorem 5.2. *Under the same conditions of Theorem 5.1, the Laplace version of Algorithm 5.1 satisfies* $(\alpha, \epsilon(\alpha))$*-RDP, where*

$$\epsilon(\alpha, \epsilon_1, \epsilon_2) = \frac{1}{\alpha - 1} \log\Bigg\{ \Big[\frac{\alpha}{2\alpha - 1} e^{\epsilon_1(\alpha-1)} + \frac{\alpha - 1}{2\alpha - 1} e^{-\epsilon_1 \alpha}\Big] \cdot \Big[\frac{\alpha}{2\alpha - 1} e^{2\epsilon_2(\alpha-1)} + \frac{\alpha - 1}{2\alpha - 1} e^{-2\epsilon_2 \alpha}\Big] \Bigg\} \tag{5.3}$$

We next show that Algorithm 5.1 with Gaussian noise also satisfies RDP.

Theorem 5.3. *Under the same conditions of Theorem 5.1, the Gaussian version of Algorithm 5.1 satisfies* $(\alpha, \epsilon(\alpha))$*-RDP, where*

$$\epsilon(\alpha, \sigma_1^2, \sigma_2^2) = \alpha(4\sigma_1^2 + \sigma_2^2)/2\sigma_1^2\sigma_2^2 \tag{5.4}$$

Proof. Let $\mathbf{v} = (v_1, ..., v_k)$ denote the output of the AboveThreshold algorithm $\mathcal{A}$, where $v_1 = ... = v_{k-1} = \perp$ and $v_k = \top$. The threshold is T for each query, and noisy threshold $\tilde{T} = T + \lambda$, where λ is a Laplace (or Gaussian) noise. Let $\nu_i, i \in [k]$ be independent Laplace (or Gaussian) noises to perturb each query result $q_i, i \in [k]$. For neighboring datasets $D \sim D'$, we have $|q_i(D) - q_i(D')| \leq \Delta_q$ for $i \in [k]$. Now consider output distributions of $\mathcal{A}$ on D and D' as $\mathbb{P}(\mathbf{v}, D)$ and $\mathbb{P}(\mathbf{v}, D')$:

$$
\begin{aligned}
\mathbb{P}(\mathbf{v}; D) &= \mathbb{P}\big[\mathcal{A}(D) = \mathbf{v}\big] \\
&= \int \dots \int \prod_{i=1}^{k-1} \mathbb{P}\big[q_i(D) + \nu_i < \tilde{T} | \tilde{T}\big] \mathbb{P}\big[q_k(D) + \nu_k \geq \tilde{T} | \tilde{T}\big] \mathbb{P}\big[\tilde{T} | T\big] \, \mathrm{d}\nu_1 \dots \mathrm{d}\nu_k \, \mathrm{d}\lambda \\
&= \int \dots \int \prod_{i=1}^{k-1} \mathbf{1}_{q_i(D)+\nu_i<\lambda} \mathbf{1}_{q_k(D)+\nu_k \geq \lambda} \, \mathrm{d}\nu_1 \dots \mathrm{d}\nu_k \, \mathrm{d}\lambda \\
&= \int \dots \int \prod_{i=1}^{k-1} \mathbf{1}_{q_i(D)+y_i<z} \mathbb{P}[\nu_i = y_i] \mathbf{1}_{q_k(D)+y_k \geq z} \mathbb{P}[\nu_k = y_k] \mathbb{P}[\lambda = z] \, \mathrm{d}y_1 \dots \mathrm{d}y_k \, \mathrm{d}z \\
&\leq \int \dots \int \prod_{i=1}^{k-1} \mathbf{1}_{q_i(D')+y_i<z+\Delta_q} \mathbb{P}[\nu_i = y_i] \mathbf{1}_{q_k(D)+y_k \geq z} \mathbb{P}[\nu_k = y_k] \mathbb{P}[\lambda = z] \, \mathrm{d}y_1 \dots \mathrm{d}y_k \, \mathrm{d}z
\end{aligned}
$$

The last step holds because $|q_i(D) - q_i(D')| \leq \Delta_q$, since $q_i(D) + y_i < z$, if $q(D) \geq q(D')$ then $q_i(D') + y_i < z$; if $q(D) < q(D')$ then $q_i(D') + y_i < z + \Delta_q$. So $q_i(D') + y_i$ is upper bounded by $z + \Delta_q$. Now we make a change of variable $z' = z + \Delta_q$, it yields

$$
\begin{aligned}
\dots &= \int \dots \int \prod_{i=1}^{k-1} \mathbf{1}_{q_i(D')+y_i<z'} \mathbb{P}[\nu_i = y_i] \mathbf{1}_{q_k(D)+y_k \geq z'-\Delta_q} \mathbb{P}[\nu_k = y_k] \mathbb{P}[\lambda = z' - \Delta_q] \left| \frac{\mathrm{d}z'}{\mathrm{d}z} \right| \mathrm{d}y_1 \dots \mathrm{d}y_k \, \mathrm{d}z \\
&= \int \dots \int \prod_{i=1}^{k-1} \mathbf{1}_{q_i(D')+y_i<z} \mathbb{P}[\nu_i = y_i] \, \mathrm{d}y_i \mathbf{1}_{q_k(D)+y_k \geq z-\Delta_q} \mathbb{P}[\nu_k = y_k] \, \mathrm{d}y_k \mathbb{P}[\lambda = z - \Delta_q] \, \mathrm{d}z \\
&= \int \int \prod_{i=1}^{k-1} \mathbb{E}_{\nu_i}\big[\mathbf{1}_{q_i(D')+\nu_i<z}\big] \mathbf{1}_{q_k(D)+y_k \geq z-\Delta_q} \mathbb{P}[\nu_k = y_k] \mathbb{P}[\lambda = z - \Delta_q] \, \mathrm{d}y_k \, \mathrm{d}z \\
&\leq \int \int \prod_{i=1}^{k-1} \mathbb{E}_{\nu_i}\big[\mathbf{1}_{q_i(D')+\nu_i<z}\big] \mathbf{1}_{q_k(D')+y_k \geq z-2\Delta_q} \mathbb{P}[\nu_k = y_k] \mathbb{P}[\lambda = z - \Delta_q] \, \mathrm{d}y_k \, \mathrm{d}z \\
&= \int \int \prod_{i=1}^{k-1} \mathbb{E}_{\nu_i}\big[\mathbf{1}_{q_i(D')+\nu_i<z}\big] \mathbf{1}_{q_k(D')+y'_k \geq z} \mathbb{P}[\nu_k = y'_k - 2\Delta_q] \mathbb{P}[\lambda = z - \Delta_q] \left| \frac{\mathrm{d}y'}{\mathrm{d}y} \right| \mathrm{d}y_k \, \mathrm{d}z \\
&= \int \int \prod_{i=1}^{k-1} \mathbb{E}_{\nu_i}\big[\mathbf{1}_{q_i(D')+\nu_i<z}\big] \mathbf{1}_{q_k(D')+y_k \geq z} \mathbb{P}[\nu_k = y_k - 2\Delta_q] \mathbb{P}[\lambda = z - \Delta_q] \, \mathrm{d}y_k \, \mathrm{d}z \qquad (5.5)
\end{aligned}
$$

The last inequality holds because $|q_k(D) - q_k(D')| \leq \Delta_q$, and one can get from $q_k(D) + y_k \geq z - \Delta_q$ that $q_k(D') + y_k$ is lower bounded by $z - 2\Delta_q$. Follows it is another change of variable $y'_k = y_k + 2\Delta_q$.

For $\mathbb{P}(\mathbf{v}, D')$, we have

$$\mathbb{P}(\mathbf{v}, D') = \int \int \prod_{i=1}^{k-1} \mathbb{E}_{\nu_i}\left[\mathbf{1}_{q_i(D')+\nu_i<z}\right] \mathbf{1}_{q_k(D')+y_k \geq z} \mathbb{P}[\nu_k = y_k]\mathbb{P}[\lambda = z]\, \mathrm{d}y_k\, \mathrm{d}z \tag{5.6}$$

Let $\Lambda(x; \mu, \lambda)$ denote the pdf of the Laplace distribution with mean μ and scale λ; let $f(x; \mu, \sigma^2)$ denote the pdf of the Gaussian distribution with mean μ and variance σ^2. For convenience, define H_α for two probability distributions P and Q with the same support as $H_\alpha := \mathbb{E}_{x\sim Q}[(P(x)/Q(x))^\alpha]$. One can solve and find that

$$\begin{aligned} H_\alpha\big(Lap(0,\lambda)\|Lap(\mu,\lambda)\big) &= \int \Lambda(x;0,\lambda)^\alpha \Lambda(x;\mu,\lambda)^{1-\alpha}\, \mathrm{d}x \\ &= \frac{\alpha}{2\alpha-1}\exp\big(\frac{\mu(\alpha-1)}{\lambda}\big) + \frac{\alpha-1}{2\alpha-1}\exp\big(\frac{-\mu\alpha}{\lambda}\big) \end{aligned}$$

$$\begin{aligned} H_\alpha\big(\mathcal{N}(0,\sigma^2)\|\mathcal{N}(\mu,\sigma^2)\big) &= \int f(x;0,\sigma^2)^\alpha f(x;\mu,\sigma^2)^{1-\alpha}\, \mathrm{d}x \\ &= \exp\Big(\frac{\alpha(\alpha-1)\mu^2}{2\sigma^2}\Big) \end{aligned}$$

For Theorem 5.2, use (5.5) and (5.6),

$$\begin{aligned} &H_\alpha\big(\mathbb{P}(\mathbf{v}, D)\|\mathbb{P}(\mathbf{v}, D')\big) = \mathbb{E}_{\mathbf{v}\sim\mathcal{A}(D')}[(\mathbb{P}(\mathbf{v}, D)/\mathbb{P}(\mathbf{v}, D'))^\alpha] \\ =& \int \dots \int \\ &\left(\prod_{i=1}^{k-1} \mathbf{1}_{q_i(D)+y_i<z}\mathbb{P}[\nu_i = y_i]\mathbf{1}_{q_k(D)+y_k\geq z}\mathbb{P}[\nu_k = y_k]\mathbb{P}[\lambda = z]\right)^\alpha \\ &\left(\prod_{i=1}^{k-1} \mathbf{1}_{q_i(D')+y_i<z}\mathbb{P}[\nu_i = y_i]\mathbf{1}_{q_k(D')+y_k\geq z}\mathbb{P}[\nu_k = y_k]\mathbb{P}[\lambda = z]\right)^{1-\alpha} \\ &\mathrm{d}y_1 \dots \mathrm{d}y_k\, \mathrm{d}z \\ \leq& \int \dots \int \prod_{i=1}^{k-1} \mathbf{1}_{q_i(D')+y_i<z}\mathbb{P}[\nu_i = y_i]\mathbf{1}_{q_k(D')+y_k\geq z} \\ &\big[\Lambda\big(y_k; 0, \frac{\Delta_q}{\epsilon_2}\big)^\alpha \Lambda\big(y_k; 2\Delta_q, \frac{\Delta_q}{\epsilon_2}\big)^{1-\alpha}\big] \\ &\big[\Lambda\big(y_k; 0, \frac{\Delta_q}{\epsilon_1}\big)^\alpha \Lambda\big(y_k; \Delta_q, \frac{\Delta_q}{\epsilon_1}\big)^{1-\alpha}\big]\, \mathrm{d}y_1 \dots \mathrm{d}y_k\, \mathrm{d}z \end{aligned}$$

$$
\begin{aligned}
&= \int\int \mathbb{E}_{\nu_i}\big[\mathbf{1}_{q_i(D')+\nu_i<z}\big]\mathbf{1}_{q_k(D')+y_k\geq z} \\
&\quad \big[\Lambda\big(y_k;0,\frac{\Delta_q}{\epsilon_2}\big)^{\alpha}\Lambda\big(y_k;2\Delta_q,\frac{\Delta_q}{\epsilon_2}\big)^{1-\alpha}\big] \\
&\quad \big[\Lambda\big(y_k;0,\frac{\Delta_q}{\epsilon_1}\big)^{\alpha}\Lambda\big(y_k;\Delta_q,\frac{\Delta_q}{\epsilon_1}\big)^{1-\alpha}\big]\,\mathrm{d}y_k\,\mathrm{d}z \\
&= \big[\frac{\alpha}{2\alpha-1}\exp\big(\frac{2\Delta_q(\alpha-1)}{\Delta_q/\epsilon_2}\big)+\frac{\alpha-1}{2\alpha-1}\exp\big(\frac{-2\Delta_q\alpha}{\Delta_q/\epsilon_2}\big)\big]\cdot \\
&\quad \big[\frac{\alpha}{2\alpha-1}\exp\big(\frac{\Delta_q(\alpha-1)}{\Delta_q/\epsilon_1}\big)+\frac{\alpha-1}{2\alpha-1}\exp\big(\frac{-\Delta_q\alpha}{\Delta_q/\epsilon_1}\big)\big]
\end{aligned}
$$

which yields the result of Theorem 5.2.

For Theorem 5.3,

$$
\begin{aligned}
&H_\alpha\big(\mathbb{P}(\mathbf{v},D)\|\mathbb{P}(\mathbf{v},D')\big) = \mathbb{E}_{\mathbf{v}\sim\mathcal{A}(D')}\left[\left(\frac{\mathbb{P}(\mathbf{v},D)}{\mathbb{P}(\mathbf{v},D')}\right)^{\alpha}\right] \\
&\leq \int\ldots\int\prod_{i=1}^{k-1}\mathbf{1}_{q_i(D')+y_i<z}\mathbb{P}[\nu_i=y_i]\mathbf{1}_{q_k(D')+y_k\geq z} \\
&\quad \big[f\big(y_k;0,\Delta_q^2\sigma_2^2\big)^{\alpha}f\big(y_k;2\Delta_q,\Delta_q^2\sigma_2^2\big)^{1-\alpha}\big]\cdot \\
&\quad \big[f\big(y_k;0,\Delta_q^2\sigma_1^2\big)^{\alpha}f\big(y_k;\Delta_q,\Delta_q^2\sigma_1^2\big)^{1-\alpha}\big]\,\mathrm{d}y_1\ldots\mathrm{d}y_k\,\mathrm{d}z \\
&= \int\int \mathbb{E}_{\nu_i}\big[\mathbf{1}_{q_i(D')+\nu_i<z}\big]\mathbf{1}_{q_k(D')+y_k\geq z} \\
&\quad \big[f\big(y_k;0,\Delta_q^2\sigma_2^2\big)^{\alpha}f\big(y_k;2\Delta_q,\Delta_q^2\sigma_2^2\big)^{1-\alpha}\big]\cdot \\
&\quad \big[f\big(y_k;0,\Delta_q^2\sigma_1^2\big)^{\alpha}f\big(y_k;\Delta_q,\Delta_q^2\sigma_1^2\big)^{1-\alpha}\big]\,\mathrm{d}y_k\,\mathrm{d}z \\
&= \exp\left(\frac{\alpha(\alpha-1)\Delta_q^2}{2}\big(\frac{4}{\Delta_q^2\sigma_2^2}+\frac{1}{\Delta_q^2\sigma_1^2}\big)\right)
\end{aligned}
$$

which yields the result of Theorem 5.3. □

One can easily verify from (5.4) that, when only one privacy parameter ρ is given, running Gaussian version of NoisyBTLS with $\sigma_1^2 \leftarrow 3/(2\rho), \sigma_2^2 \leftarrow 3/\rho$ would satisfy $(\alpha, \alpha\rho)$-RDP.

5.5.2 Private Backtracking Line Search Based Stochastic Gradient Descent

Now we present our main algorithm, called Differentially Private Backtracking Line Search-based Stochastic Gradient Descent (DP-BLSGD). Algorithm 5.2 shows the pseudocode.

Starting with initial parameter vector $\mathbf{w}_0$, at iteration t, the algorithm evaluates the gradient $\nabla f(\mathbf{w}_t)$ over a minibatch B. To bound the sensitivity, it applies the gradient clipping [14]. Specifically, it computes the per-example gradient $\mathbf{g}_i = \nabla f(\mathbf{w}_t, \mathbf{d}_i)$ for each $\mathbf{d}_i \in B$ and applies the clipping function to $\mathbf{g}_i$, i.e., $\mathsf{clip}(\mathbf{g}_i, C_{grad})$. The clipping function is defined as

$$\mathsf{clip}(\mathbf{g}, C) = \frac{\mathbf{g}}{\max(1, \|\mathbf{g}\|_2/C)}\,. \tag{5.7}$$

The application of clipping function in line 5 ensures that the L_2 norm of every per-example gradient in the summation is no greater than the threshold C_{grad}, and hence it bounds the L_2 sensitivity of summed gradient to C_{grad}. After summing the clipped per-example gradients, it adds Gaussian noise with variance $C_{grad}^2/2\rho_{grad}$ to each coordinate.

The step size η for iteration t is computed by calling the NoisyBTLS function with passing noisy gradient $\tilde{\mathbf{g}}_t$ as input. When the step size η returned by NoisyBTLS is greater than 0, the algorithm performs SGD update in line 13.

We now discuss how the proposed algorithm dynamically adjusts the privacy budget to account for the case in which the algorithm fails to find a reasonably large step size.

Privacy budget adaptation When NoisyBTLS fails to find a step size within `max_it` iterations, there are two possibilities. First, the current privacy budget ρ_{grad} assigned for evaluating the gradient is too small that noise dominates the gradient. The remedy for this case is to increase the privacy budget. The second possibility is that ρ_{grad} is large enough not to remove the gradient signal but the large noise in NoisyBTLS prevents it from finding the step size satisfying the condition (5.2). In this case, we need a more accurate measurement of gradient and an increased privacy budget for the backtracking line search. To distinguish these two case, DP-BLSGD maintains the moving average of angles between two

Algorithm 5.2 Rényi Differentially Private Backtracking Line Search Based Sub-sampled Gradient Descent (DP-BLSGD)

Input: Dataset $D = \{\mathbf{d}_1, ..., \mathbf{d}_n\}$, loss function $F(\mathbf{w}, D)$, clipping thresholds C_{obj}, C_{grad}, sampling ratio q, privacy budget for line search ϵ_{BT}, privacy budget for gradient ρ_{grad}, budget increase rate ξ, initial learning rate η_0, total privacy budget $\epsilon_{tot}(\alpha)$.
Initialize $\mathbf{w}_0$ randomly. $\overline{\theta} \leftarrow 90°$.
for $t = 0, 1, ...$ **do**
    Sample a mini-batch B by sampling ratio q
    $\mathbf{g}_t \leftarrow \sum_{\mathbf{d}_i \in B} \mathsf{clip}(\nabla f_i(\mathbf{w}_t, \mathbf{d}_i), C_{grad})$ ▷ see (5.7)
    $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{|B|}(\mathbf{g}_t + \gamma)$, where $\gamma \sim \mathcal{N}(0, (C_{grad}^2/2\rho_{grad})\mathbb{I})$
    $\epsilon_{tot}(\alpha) \leftarrow \epsilon_{tot}(\alpha) - \mathsf{amp}(\alpha\rho_{grad})$ ▷ see 5.5.3
    $\eta \leftarrow 0, \epsilon_{ls}(\alpha) \leftarrow \epsilon(\alpha, \epsilon_{BT}/2, \epsilon_{BT}/4))$ ▷ see (5.3)
    while $\eta = 0$ and $\epsilon_{tot}(\alpha) > \epsilon_{ls}(\alpha)$ **do**
        $\eta \leftarrow \text{NoisyBTLS}(f, B, \mathbf{w}_t, \tilde{\mathbf{g}}_t, \eta_0, \epsilon_{BT})$
        $\epsilon_{tot}(\alpha) \leftarrow \epsilon_{tot}(\alpha) - \mathsf{amp}(\epsilon_{ls}(\alpha))$ ▷ see 5.5.3
        if $\eta > 0$ **then**
            $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta\tilde{\mathbf{g}}_t$
            Update θ_t and $\overline{\theta}$ according to (5.8)
        else if $\epsilon_{tot}(\alpha) > \alpha\rho_{grad}$ **then**
            $\epsilon_{tot}(\alpha) \leftarrow \epsilon_{tot}(\alpha) - \mathsf{amp}(\alpha\rho_{grad})$ ▷ see 5.5.3
            $\rho_{grad}, \epsilon_{BT}, \tilde{\mathbf{g}}_t \leftarrow \text{CheB}(\rho_{grad}, \epsilon_{BT}, \xi, \tilde{\mathbf{g}}_t)$
        else
            break
        end if
        $\epsilon_{ls}(\alpha) \leftarrow \epsilon(\alpha, \epsilon_{BT}/2, \epsilon_{BT}/4))$ ▷ see (5.3)
    end while
    if $\epsilon_{tot}(\alpha) \leq \alpha\rho_{grad}$ **then**
        break
    end if
end for
Output: $\mathbf{w}_{t+1}$

Algorithm 5.3 Check and Enlarge Budget for SGD (CHEB)

1: **Input**: current budget ρ_{grad}, ϵ_{BT}, budget increase rate ξ, perturbed gradient $\tilde{\mathbf{g}}_t$.
2: Sample a mini-batch B by sampling ratio q
3: $\mathbf{g}_{t2} \leftarrow \sum_{\mathbf{d}_i \in B} \left(\nabla f(\mathbf{w}_t; \mathbf{d}_i) / \max(1, \frac{\|\nabla f(\mathbf{w}_t; \mathbf{d}_i)\|_2)}{C_{grad}}) \right)$
4: $\tilde{\mathbf{g}}_{t2} \leftarrow \frac{1}{|B|}(\mathbf{g}_{t2} + \gamma_2)$, where $\gamma_2 \sim \mathcal{N}(0, (C_{grad}^2 / 2\rho_{grad})\mathbb{I})$
5: $\theta \leftarrow$ AngleBetween$(\tilde{\mathbf{g}}_t, \tilde{\mathbf{g}}_{t2})$
6: Calculate θ_{max} and θ_{min} ▷ see (5.9)
7: **if** $\tilde{\mathbf{g}}_t \cdot \tilde{\mathbf{g}}_{t2} < 0$ or $\theta > \theta_{max}$ **then**
8: $\quad \rho_{grad} \leftarrow (1 + \xi)\rho_{grad}$
9: **else if** $\theta < \theta_{min}$ **then**
10: $\quad \epsilon_{BT} \leftarrow (1 + \xi)\epsilon_{BT}$
11: **end if**
12: $\tilde{\mathbf{g}}_t \leftarrow (\tilde{\mathbf{g}}_t + \tilde{\mathbf{g}}_{t2})/2$
13: **Output**: $\rho_{grad}, \epsilon_{BT}, \tilde{\mathbf{g}}_t$

consecutive gradients, and it is updated at every iteration (line 14 in Algorithm 5.2) as follows:

$$\begin{aligned} \theta_t &\leftarrow \text{AngleBetween}(\tilde{\mathbf{g}}_t, \tilde{\mathbf{g}}_{t-1}) \\ \overline{\theta} &\leftarrow \psi\overline{\theta} + (1 - \psi)\theta_t \,, \end{aligned} \tag{5.8}$$

where $\psi \in (0, 1)$ is a parameter controlling the decay rate of old information. Note that $\overline{\theta}$ is initialized to $90°$ for the first iteration and line 14 is not executed when $t = 0$. When $\eta = 0$ is returned by NoisyBTLS, the algorithm evaluates another gradient $\tilde{\mathbf{g}}_{t2}$ using the budget of ρ_{grad} and measures the angle θ between $\tilde{\mathbf{g}}_t$ and $\tilde{\mathbf{g}}_{t2}$ (line 5 in Algorithm 5.3). If θ is greater than the moving average-based threshold θ_{max}, the algorithm increases the privacy budget ρ_{grad} for gradient computation. When θ is smaller than the minimum threshold θ_{min}, it indicates that the search might fail because the privacy budget ϵ_{BT} assigned for noisy backtracking line search (i.e., Algorithm 5.1) is too small. Hence, we increase ϵ_{BT} in this case. The threshold values θ_{max} and θ_{min} are calculated as follows:

$$\theta_{max} \leftarrow \phi_{max} \times \overline{\theta} \,, \quad \theta_{min} \leftarrow \phi_{min} \times \overline{\theta} \,, \tag{5.9}$$

where $\phi_{max} > 1$ and $0 < \phi_{min} < 1$ are hyper-parameters. Empirically, we observe this budget adaptation strategy is especially effective for convex optimization problems.

Intelligent backtracking To reduce the number of times the algorithm redundantly backtracks due to unnecessarily large initial step size η_0, DP-BLSGD maintains a list Ω of previously selected step sizes. After every τ iterations, η_0 is updated as $\eta_0 \leftarrow \min\{\varsigma \cdot \max\{\Omega\}, \eta_0\}$, and Ω is reset to an empty set. The parameter $\varsigma > 1$ guarantees the line search starts with sufficiently large initial step size but not too large to avoid redundant backtrackings. In our experiments in Section 5.6, we set $\varsigma = 1.2$. Note that, in a non-private setting, the line search algorithm proposed in [74] resets the initial step size in a similar way, but it simply resets η_0 to a multiple of η selected in the previous iteration. However, in a private setting, this strategy of resetting η_0 at every iteration can make the search unstable as step sizes selected by a noisy backtracking algorithm can fluctuate due to noise.

Theorem 5.4. *Algorithm 5.2 satisfies RDP.*

Proof. As shown in Section 5.5.3, since each data-dependent step is RDP, one can track the privacy budget at each iteration (Theorem 5.2 or 5.3 for NoisyBTLS; noisy gradient is $(\alpha, \alpha\rho_{grad})$-RDP), then use Lemma 2.4 to amplify for subsampling, and Lemma 2.2 to calculate the overall privacy. □

5.5.3 Privacy Budget Tracking

Each sub-routine in the proposed algorithm incurs different amount of privacy loss. To ensure that the total privacy budget spent by the algorithm is smaller than the given total privacy budget $\epsilon_{tot}(\alpha)$ (so that the entire algorithm satisfies $(\alpha, \epsilon_{tot}(\alpha))$-RDP), the algorithm computes the total privacy loss incurred by each function call under RDP framework. Specifically, it computes the amount of required privacy budget using Theorem 5.2 and Lemma 2.1, followed by privacy amplification through Lemma 2.4 (denoted in Algorithm 5.2 as amp). Following the RDP composition (Lemma 2.2), the algorithm subtracts it from the running privacy budget before calling each sub-routine. Recall that in RDP the privacy loss is a function of α (the order of Rényi divergence). In a practical implementation of the algorithm to satisfy (ϵ', δ)-DP, one can first calculate $(\alpha, \epsilon(\alpha))$ for a series of α values (e.g., integers between 2 and 500) which satisfy that (ϵ', δ)-DP by Proposition 2.1, maintaining and keep track of total privacy budget spent for each case of α, and halt and return the result when budget $\epsilon_{tot}(\alpha)$ for all α values are lower than the minimum budget required to call a sub-routine.

5.5.4 Clipping Threshold Adaptation

The gradient and objective clipping techniques allow to effectively bound the sensitivity but, if they are used with incorrectly chosen threshold values, they can degrade the utility. For example, if C_{grad} is too high but the norm of gradient is small, it is likely that the noise dominates the gradient due to high sensitivity. On the other hand, if C_{grad} is too small, then it clips out useful information. During the model training, the norm of gradients decreases as the parameter vector $\mathbf{w}_t$ gets closer to the optimal values, and hence a large clipping threshold might not be necessary in the later stage of training. Motivated by this, we propose to adaptively decrease the clipping threshold C_{grad} and C_{obj} if the algorithm decides to increase ρ_{grad} (line 8 in Algorithm 5.3) during a single SGD update. The algorithm decreases the threshold only once per each SGD update regardless of how many times ρ_{grad} is increased.

$$C_{grad} \leftarrow (1-\zeta)C_{grad}\,, \quad C_{obj} \leftarrow (1-\zeta)C_{obj}\,, \tag{5.10}$$

where ζ is a hyperparameter that determines the rate of decrease. The proposed algorithm with the clipping threshold adaptation is called DP-BLSGD-AC. Note that this strategy does not require any extra privacy budget since the condition is based on privately released information.

5.6 Experimental Results

5.6.1 Models

We evaluate the performance of proposed algorithm on both convex and nonconvex problems. For convex problems, we consider training two models: logistic regression and linear SVM. Let $\mathbf{d}_i = (\mathbf{x}_i, y_i)$, for $i = 1, \ldots, n$, where $\mathbf{x}_i$ is a feature vector and $y_i \in \{-1, +1\}$ is its label. The objective function of logistic regression is

$$F(\mathbf{w}; D) := \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))\,,$$

which is convex and smooth. For SVM, we use the hinge loss, which is convex but non-smooth, and

$$F(\mathbf{w}; D) := \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)\,.$$

Table 5.2: Summary of datasets

Dataset	Size	Dimension	Baseline
Adult	$48,842$	124	0.761
Bank	$45,211$	33	0.883
IPUMS-BR	$38,000$	53	0.507
IPUMS-US	$40,000$	58	0.513
MNIST	$60,000/10,000$	$1 \times 28 \times 28$	~ 0.1
FMNIST	$60,000/10,000$	$1 \times 28 \times 28$	~ 0.1
Cifar-10	$50,000/10,000$	$3 \times 32 \times 32$	~ 0.1

We also apply our algorithms on non-convex problems, training neural networks for image classification tasks. We trained the neural networks with two different architectures: multi-layer perception (MLP) and convolution neural network (CNN). The details of network architectures are discussed in Section 5.6.7. To avoid overfitting, we applied a L_2 regularization on the model parameters with a coefficient $\mu = 0.001$ for all models.

5.6.2 Datasets and Pre-proessing

A summary of all datasets used in our experiments is shown in Table 5.2. Four census datasets were used for convex optimization: Adult [31], Bank [82], IPUMS-BR, and IPUMS-US [33]. All categorical attributes are pre-processed by one-hot encoding, and numeric ones are scaled to [0, 1]. Three datasets were used for training neural networks: MNIST, FMNIST, and Cifar-10. Note that these three datasets have separate dataset for testing. For other datasets, we report the averaged performance of 10-fold cross validation. MNIST and Fashion MNIST (FMNIST) datasets contain gray-scale images, while Cifar-10 dataset consists of RGB images. Each pixel in each channel is re-scaled into [-1, 1]. We run each experiment 5 times and report the averaged performance.

5.6.3 Baselines

For convex problems, we compare the performance of our proposed algorithms, DP-BLSGD, DP-BLGD, and DP-BLSGD-AC, with 8 baseline algorithms: DP-AGD [17], DP-SGD [14], OUTPERT-

RSGD [10], OUTPERT-GD [8], OBJPERT [7,12], PrivGene [83], MAJORITY, and NON-PRIVATE. DP-AGD is the adaptive full-batch gradient descent algorithm, which selects step size by NOISYMIN. DP-BLGD is the batch gradient descent version of our DP-BLSGD, which uses the budget increasing technique of DP-AGD. DP-SGD is the gradient perturbation algorithm presented in [14]. OUTPERT-RSGD and OUTPERT-GD are both output perturbation algorithms, the former calculates sensitivity based on permuted SGD with averaging, and the latter calculates sensitivity depending on batch gradient descent. OBJPERT is the objective perturbation algorithm which inject noise into loss function. PRIVGENE is private model fitting based on genetic algorithm. NON-PRIVATE is the non-private baseline, which uses L-BFGS to search for an optimal solution. MAJORITY classifies every sample as the major class. For the baseline algorithms which require smoothness of the loss function, the SVM experiments are performed on Huberized SVM, where

$$F(\mathbf{w}, D) := \frac{1}{n}\sum_{i=1}^{n}\begin{cases} 1 - y_i\mathbf{w}^T\mathbf{x}_i & \text{if } y_i\mathbf{w}^T\mathbf{x}_i < 1-\hbar \\ \frac{1}{4\hbar}(1+\hbar-y_i\mathbf{w}^T\mathbf{x}_i)^2 & \text{if } |1-y_i\mathbf{w}^T\mathbf{x}_i| \le \hbar \\ 0 & \text{otherwise} \end{cases}$$

and $\hbar$ is a hyperparameter set to 0.5.

For neural network models, we compare our algorithms with the gradient perturbation algorithms proposed in [14], which injects Gaussian noise into the sub-sampled clipped gradients, for both SGD and Adam versions. The NON-PRIVATE baseline shows the performance of ADAM optimizer.

5.6.4 Hyperparameter setting

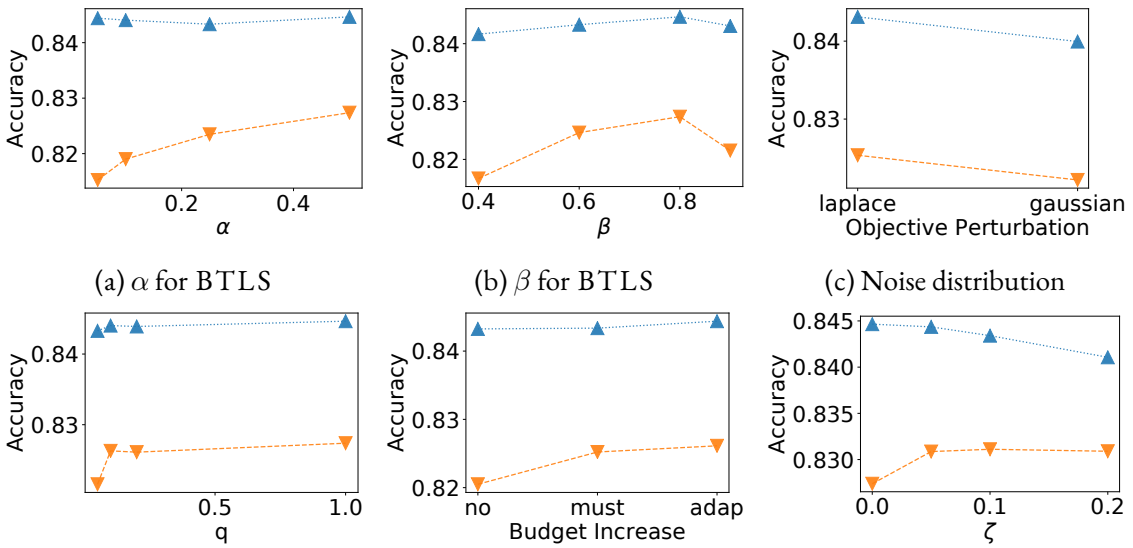
We fix $\delta = 10^{-8}$ for all experiments. In order to make fair comparisons between algorithms, we convert RDP into (ϵ, δ)-DP using the conversion tool given in Proposition 2.1. For convex models, we set the hyperparameters as follows. To initialize the initial privacy budget ϵ_{BT} and ρ_{grad}, we heuristically determine a per-iteration budget as $\epsilon_{iter} = \epsilon/(2 \times 50)$. The intuition behind this setting is that we expect the algorithm would approximately require 50 iterations. Given the per-iteration ϵ_{iter}, we set $\epsilon_{BT} = \epsilon_{iter}$ and $\rho_{grad} = \frac{1}{2}\epsilon_{iter}^2$.

The sampling rate is set to $q = 0.1$, and the clipping thresholds are set as $C_{grad} = 3$ and $C_{obj} = 1$ for all gradient perturbation algorithms. The privacy budget increase parameter is set as $\xi = 0.3$. The

hyperparameters of NoisyBTLS algorithm are set as follows: $\alpha = 0.5, \beta = 0.8, \tau = 10, \varsigma = 1.2, \phi_{max} = 1.1, \phi_{min} = 0.5$, and $\psi = 0.8$. For DP-BLSGD-AC, we set the clipping threshold decrease rate parameter $\zeta = 0.05$. Hyperparameters of the baseline algorithms are set as suggested in their papers.

For neural network models, we set subsampling rate $q = 1/200$, $C_{grad} = 3$, and $C_{obj} = 3$. Since all the algorithms being compared can achieve RDP and they are all gradient perturbation-based ones, we plot the performance over iterations, and use RDP for composition of mechanisms. We set the hyperparameters of NoisyBTLS as $\alpha = 0.001$ and $\beta = 0.8$. The reason is that for over-parameterized models we empirically observed that setting α to small helps fasten training. For DP-Adam, we use the default parameter settings. For DP-SGD, after tuning, we set $\eta = 0.2$ for MNIST and FMNIST models, and $\eta = 0.1$ for Cifar-10 models.

5.6.5 Effect of Hyperparameters



(a) α for BTLS (b) β for BTLS (c) Noise distribution

(d) Sampling ratio q (e) Budget allocation strategy (f) Clipping threshold decay rate ζ

Figure 5.1: Impact of hyperparameters on the performance (▲: $\epsilon = 0.2$, ▼: $\epsilon = 0.05$)

We evaluate the effect of 6 important hyperparameters of our algorithms: α, β used for backtracking line search, noise distribution (Laplace or Gaussin), sampling ratio q, budget allocation strategy, and clipping threshold decay rate ζ. Figure 5.1 shows the effects of varying hyperparameters on the performance of logistic regression model. For this experiment, Adult dataset was used. As shown in Figure 5.1a and 5.1b,

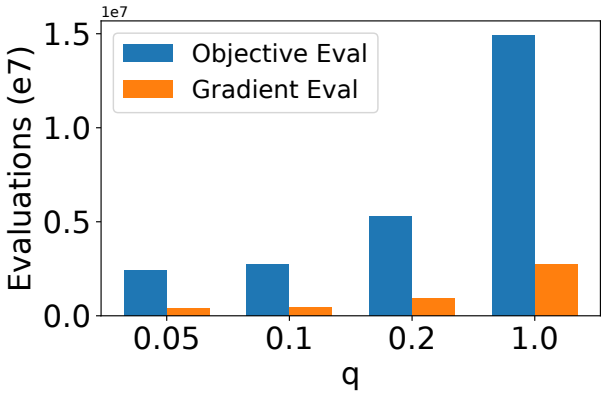


Figure 5.2: Number of gradient and objective evaluations for different sampling ratio at $\epsilon = 0.4$

the algorithm shows relatively robust performance against the choice of α and β when $\epsilon = 0.2$. When $\epsilon = 0.05$, the algorithm achieved its best performance at $\alpha = 0.5$ and $\beta = 0.8$. This is because choosing smaller step sizes allows the algorithm to control the variance due to noise. Larger values of α will encourage the algorithm to choose small step size by setting the expected reduction in the objective high. Small β would give the algorithm a large jump of candidate step size each time, therefore generally $\beta = 0.8$ is a suitable choice. The proposed algorithm achieved slightly higher accuracy when the noise for backtracking line search was drawn from the Laplace distribution. When $\epsilon = 0.2$, the algorithm is robust to the choice of sampling ratio q, but q cannot be too small at high privacy level, since gradients calculated on smaller batches have higher variance. Although the performance is similar for stochastic and full gradient descent, as Figure 5.2 shows, subsampling can greatly reduce the number of objective and gradient evaluations. For budget allocation mechanisms for SGD, "no" means never increase budget, "must" means always increase budget regardless of angle measurement, and "adap" means adaptively increase budget based on angle measurement. We can see the angle measurement is indeed beneficial. It is hard to determine the effect of adaptively decreasing clipping threshold, since the results show that for $\epsilon = 0.05$, it benefits the performance, but for $\epsilon = 0.2$ it does not. Therefore in the next section we plot performance of BLSGD with and without adaptive clipping.

5.6.6 Performance of Convex Optimization

Figure 5.3 and Figure 5.4 plots the testing data accuracy (top) and objective values (bottom) of the algorithms against the privacy parameter ϵ, for logistic regression and SVM, respectively. For the algorithms we proposed, we show results using Laplace version of NoisyBTLS, since it slightly outperforms the one using Gaussian version. DP-BLSGD, DP-BLSGD-AC, and its full-batch version (DP-BLGD) outperform the baseline algorithms in most cases. They outperform the DP-AGD algorithm, which shows that the NoisyBTLS based technique performs better than the NoisyMin based step size selection. Since the DP-BLGD applies the same budget increasing mechanism as DP-AGD, and both algorithms use full gradient descent, it shows that the improvement of DP-BLGD over DP-AGD is a result of Armijo line search technique. Our algorithms also outperform the state-of-the-art output perturbation algorithm, Outpert-RSGD, on 3 out of 4 datasets. Objpert and DP-SGD show low performance when ϵ is small. This indicates that step size selection and adaptive budget control are useful tools to achieve a high privacy level. The Bank dataset is an exception, which Outpert-RSGD outperforms our algorithms. But this dataset has very small training range since the majority and non-private baselines are very close, which might affect the performance of gradient perturbation based algorithms. When clipping threshold adaptation is applied (DP-BLSGD-AC), the performance can be slightly increased in some datasets.

5.6.7 Performance of Neural Network models

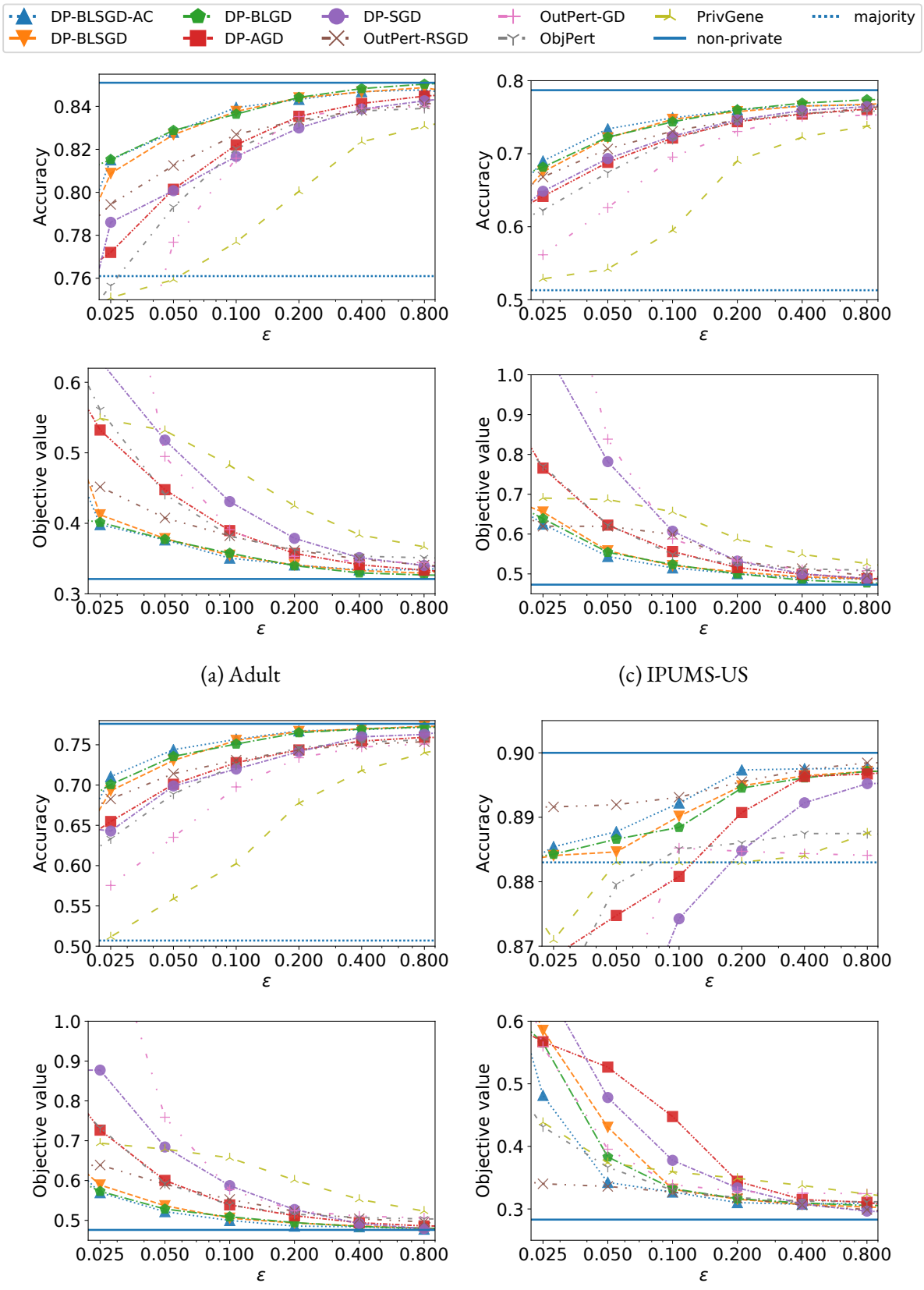
For MLP, we have one hidden layer with 1000 units for MNIST, and 2 hidden layers with 256 units each for FMNIST; for CNN on MNIST and FMNIST, we stack a convolutional layer with 6 output channels, a max pooling layer, another convolutional layer with 16 output channels, another max pooling layer, and 2 fully connected layer with width 256 and 128, respectively. For CNN on Cifar-10, we stack a convolutional layer with 32 out channels, another with 64 out channels, a max-pooling layer, 2 convolutional layers with 128 out channels, a max-pooling layer, 2 convolutional layers with 256 out channels, a max-pooling layer, and 3 fully connected layers with size 4096, 1024, 512, respectively. In order for our DP-BLSGD to accumulate privacy parameter ϵ at the same speed with DP-SGD and DP-Adam, we set the the NoisyBTLS to return $\beta^{\texttt{max\_it}}\eta_0$ instead of 0 if it evaluates over all the `max_it` candidates. (Thus, we let our algorithm DP-BLSGD just perform step-size selection, without budget increasing.) We also set

the same per-iteration budget ρ_{iter} for three algorithms: for DP-SGD and DP-Adam, we use ρ_{iter} to determine noisy scale σ^2; for DP-BLSGD, we use 10% of ρ_{iter} for NoisyBTLS (Gaussian version), and 90% for gradient perturbation. Therefore, each iteration is $(\alpha, \alpha\rho_{iter})$-RDP for all three algorithms, and it would be a fair comparison of performances against iterations.

The classification performance for neural network models are shown Figure 5.5. The bottom row shows the step size selected by DP-BLSGD. As an expected behavior, it is decreasing during training. DP-BLSGD outperforms DP-SGD and DP-Adam in these aspects: For MLP networks, it can reach to a high testing accuracy in less iterations, and converge to an accuracy similar as other methods. For CNN networks on MNIST and FMNIST, it converges much faster than DP-SGD and DP-Adam, and result in lower objective values. On Cifar-10 dataset, although the gap between non-private and private algorithms is larger, DP-BLSGD still achieves better performance in less iterations compare to the two private baselines, especially in objective value. Note that, since our DP-BLSGD did not perform budget increasing for neural network models, it uses the same per-iteration budget across the training as DP-SGD and DP-Adam, so our method may still face too noisy gradients in later stage, preventing it from converging to a higher accuracy. Since neural network models needs much more iterations to train, the privacy leak would accumulates too fast if we keep increasing per-iteration budget. However, our results show that step-size selection through line search alone can help increase the performance to a certain level, and accelerate the convergence as well.

5.7 Conclusions

We presented a Rényi differentially private SGD algorithm, in which step sizes are adaptively chosen using the Armijo condition. To improve the reliability of chosen step sizes, we also introduce strategies for adaptive privacy budget allocation. Our empirical evaluations on both convex and nonconvex problems demonstrate that classical line search can help automatically set the step size and improve the utility. We also introduced practical techniques for improving the runtime adaptivity of private optimization algorithms, which allows the algorithm to accelerate by making quick progresses.



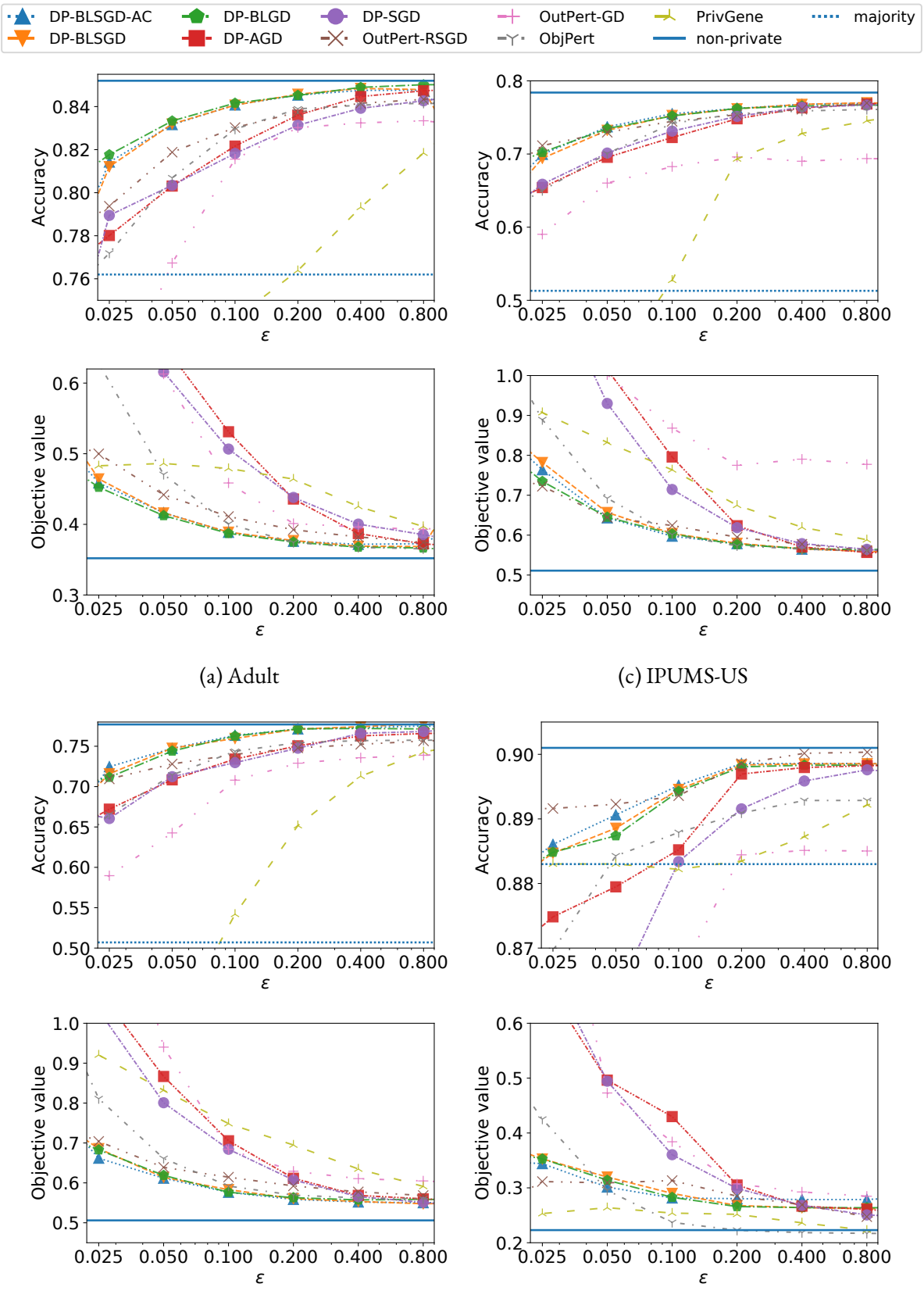
(a) Adult

(c) IPUMS-US

(b) IPUMS-BR

(d) Bank

Figure 5.3: Logistic regression result by ϵ (Top: Classification accuracy; Bottom: Objective value)



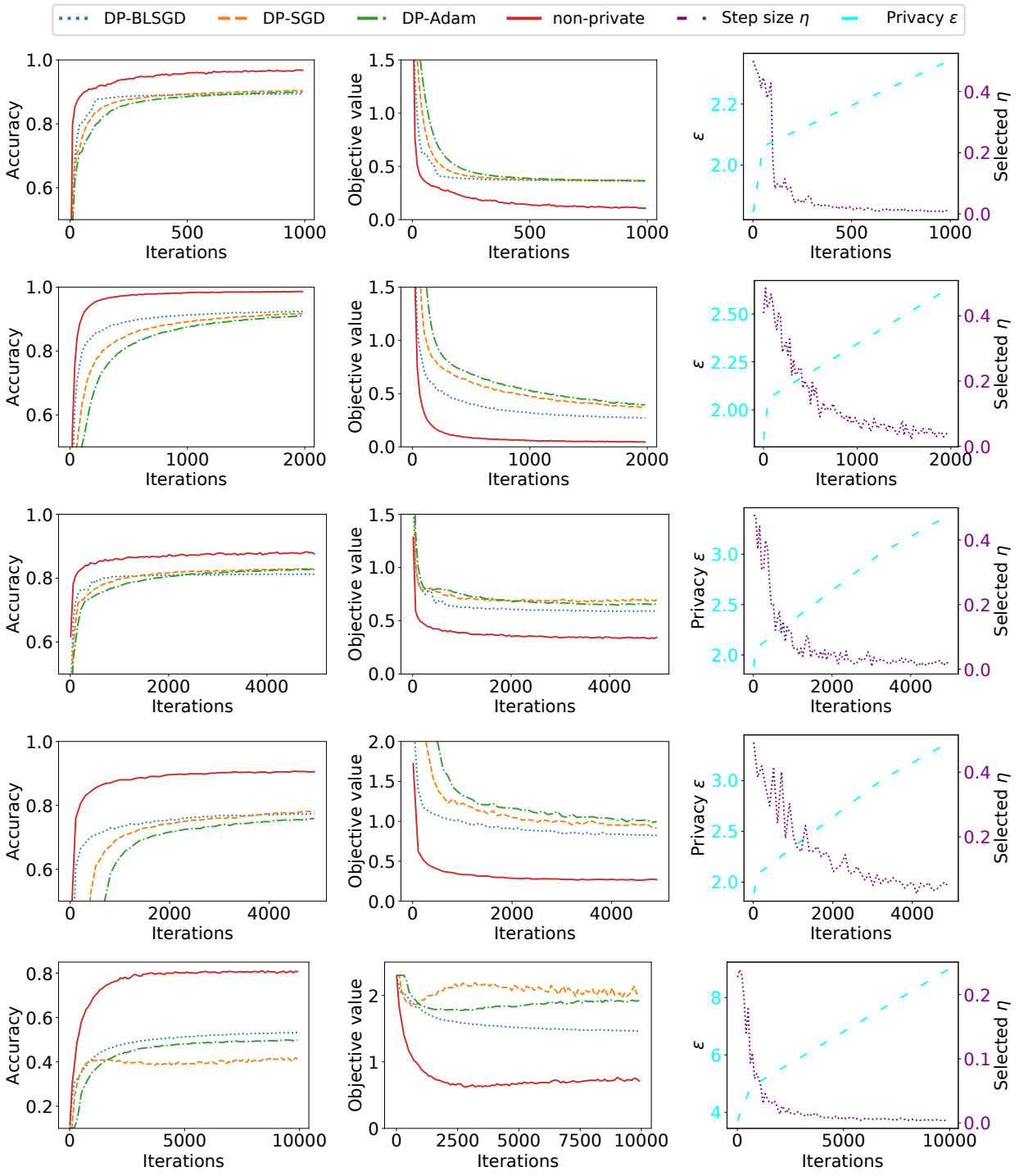
(a) Adult

(c) IPUMS-US

(b) IPUMS-BR

(d) Bank

Figure 5.4: SVM result by ϵ (Top: Classification accuracy; Bottom: Objective value)



(a) Classification Accuracy (b) Objective value (c) Privacy ϵ and selected step size

Figure 5.5: Neural network model results (Top to Bottom: MNIST MLP, MNIST CNN, FMNIST MLP, FMNIST CNN, and Cifar10 CNN)

Chapter 6

Private Ranking: Achieving High Utility Privacy for User Preference

6.1 Motivation

This chapter is a project of applying differential privacy on a realistic domain: the ranking problem. While the algorithms presented in previous chapters can be used on any empirical risk minimization problems, given that the assumptions of objective functions hold, some specific problems in the real world may be some special characteristics, either in the optimization goal or in the training data. Take the ranking problem in information retrieval domain as an example: although the learning can be formulated as an ERM problem, its training data are session based, and the information coming from one individual has multiple portions: the documents which are presented to the user, which are not exclusive to the user, and the preference of the user to the documents. In this chapter, we look into this problem closely, and present a privacy framework which can well fit the characteristic of the ranking data, with a sensitivity calculation technique based on gradient perturbation algorithm to privatise the supervised learning to rank algorithm.

6.2 Introduction

The rapid growth of usage of the Internet has significantly increased the speed of generation of web-page documents, with a growing need of intelligent information retrieval (IR) [84]. A large group of IR problems can be formulated as a ranking problem to sort the candidate documents from most to least relevant. Supervised ranking models are trained upon historical user data, where the labels assigned to the documents are considered as the ground truth to supervise the machine learning process. However, historical users' preferences upon the candidate documents represent personal information about the user, which may cause private issues if it is leaked to or inferred by adversaries. Therefore, protecting the privacy of user's information should be considered as an essential task to make a recommendation system trustworthy.

Recent progress in research in differential privacy has make it a standard to protect information generated by humans in a broader range of supervised machine learning models, such as deep learning [14]. As the *de facto* technique to protect the privacy, it provides protection and against strongest adversaries with arbitrary side information, and provides privacy-utility trade-offs for the model publisher to control the level of privacy intended.

In this chapter, we consider the problem of protecting the privacy of user data when training a ranking model. One key feature of DP is the clear definition of "neighboring" datasets, which specifies the boundaries between individuals. For this reason, we choose the listwise models, where each of the training units clearly represents an individual user who made a judgement on the documents presented to her, so that neighboring datasets can be clearly defined. But for other type of models like pointwise and pairwise, each user might correspond to multiple data rows, so it is difficult to scale the effect of one user, which correspond to multiple rows in the training data. To formally quantify the privacy protection, we present a novel privacy framework which only considers the preference label from the user as sensitive information. The document vector, on the other hand, can be considered as public information, since it is not dependent on the user's preference. Under this definition, a strong adversary would gain limited confidence on any target individual by inferring from the released model. With such assumption, document vectors can help us to calculate the sensitivity of accumulated gradients of the BayesRank model, where the users' preference information is access only through NDCG evaluation. Furthermore, while existing private SGD algorithms requires gradient clipping with a pre-defined clipping threshold, this algorithm does

not need such procedure, since NDCG is naturally bounded. Instead, we apply a spectral normalization technique to regularize the scale of the model parameters, to avoid the sensitivity going too high. We also tested the private algorithm without the assumption of feature vector being public information, by the gradient clipping and perturbation technique to train a neural network model. Experimental results shows that, improved performance is observed if we only protect the user's information, and consider feature vector as public to assist the analysis of sensitivity.

Our contributions are summarized as follows:

- We look into the characteristics of listwise ranking model, and propose a privacy framework to protect the user's preference on the documents. This framework fits the Rényi differential privacy.
- We analyze the sensitivity of the BayesRank training algorithm, and propose a way to protect the user's information during training such model.
- We evaluate the performance of our proposed algorithm, and compare its performance with a series of other privatization methods, to show effectiveness of our algorithm.

The rest of this chapter are organized as: Section 6.3 reviews the related work; Section 6.4 summarizes important definitions and lemmas used in this chapter; Section 6.5 presents the main algorithm; and experimental results are presented in Section 6.6; Section 6.7 concludes the chapter.

6.3 Related Work

The information retrieval (IR) task is considered as a ranking problem in machine learning community. This problem has extended its domain beyond document and webpage ranking into music, video, and product ranking [85, 86]. Recently progress of machine learning models for solving a ranking task utilizes supervised learning, and tries to build a model by historical user's preferences on documents as the training data. The supervised learning models can be summarized into three categories: Pointwise models formulate the ranking problem as a regression or classification problem, and pre-processing the training data into feature-label vectors of documents and associated labels (scores), assuming each vector are independently generated, such as logistic regression model in [87], maximum entropy model in [88], and McRank model in [89].

Pairwise models are also called comparison based models, which pre-process the training data into comparing pairs. It iteratively teaches the model a binary preference relationship between two pieces of data. For each session in the training data, pairs of documents with different labels assigned by one user are generated as a "pair", which is fed into the scoring function, and the training objective is to learn the preference between the two documents in comparison by defining a pairwise loss function. For example, Herbrich et al. presented the RankSVM algorithm in [90], which is improved by Joachims et al. in [91] and [92]. Burges et al. presented the RankNet algorithm [93] based on neural network, where the training objective is to minimize the mis-matching pairs in a given session. This technique was further improved by the LambdaRank in [94], which heuristically use the absolute change of NDCG ($|\Delta_{NDCG}|$) in the session to re-weight the gradients of the training pairs. Besides, pairwise boosting method was first proposed in by Freund et al. in [95], relying on gradient boost tree, and further extended by Zheng et al. through QBRank algorithm in [96], which proposed a boosted successive quadratic approximation algorithm. Compare to pointwise algorithms, pairwise algorithms take into consideration of the nature of "comparison" for a ranking model, therefore resulting in better performance in NDCG, while the comparison based training does not increase the time complexity significantly. However, due to the inconsistency of the number of comparable pairs each session can generate, the pre-processing would cause an imbalance of data pairs, which are imperfectly treated as independent samples; and pairs of data provide weak and ambiguous training information compare to what can be provided in a list.

Listwise models directly consume ranking sessions generated by historical users. It comes as a natural way of learning, since IR measures are all defined on sessions, and the dataset does not need to be pre-processed. The LambdaRank algorithm [94] utilizes NDCG measures, which relies on the session where the pair belongs to, so it can be considered as a precursor of listwise techinque. Other than that, most recent Listwise models defines and attempts to minimize a training objective as a surrogate loss function. For example, Cao et al. proposed ListNet [97], which trains a neural network based scoring function, and minimizes the cross entropy loss of two probability distributions of permutations. The RankCosine [98] method from Qin et al. defines another loss function, based on the Cosine similarity between the score vector of the ground truth and that of the predicted result, with a RankBoost [95] scoring function. Xia et al. [99] summarized those listwise methods, and proposed another surrogate loss function, ListMLE. It is arguable that by minimizing a *surrogate* loss function, the empirical risk minimization model does not directly optimize IR performance in terms of evaluations such as NDCG and MAP. For this reason, effort

has been made to optimize the evaluation measures directly, such as AdaRank [100] based on boosting tree, PermuRank [101] and SVM-MAP [102] based on SVM, and BayesRank [103] based on neural network. These methods involves the calculation of IR measurements to guide the formulation of gradients, and BayesRank tackles the ranking in a decision making perspective, trying to minimize the Bayesian risk of the truth permutation. In addition to these three categories, Fernando et al. proposed MidRank algorithm [104] on image data, which lay between pairwise and listwise approaches, learning from moderately sized sub-sequences with useful structural ranking information.

As the standard technique to protect the privacy of training data, differential privacy (DP) has been widely used in empirical risk minimization models. For first-order approximation methods such as (stochastic) gradient descent, privacy can be achieved by gradient perturbation mechanism for multiple accesses of the data through gradient calculation, and the overall privacy is measured by privacy composition lemma. Bassily et al. proposed the first gradient perturbation algorithm [13], with the "strong composition" method to account for the privacy loss over multiple iterations. Then, Abadi et al. proposed the "moment accountant" method [14], which gave a tighter bound on privacy amplification and accountant, and the sensitivity is bounded by gradient clipping. Later, Mironov et al. extended the privacy composition into Rényi differential privacy (RDP) [4], which constrains the privacy loss random variable $\log \frac{\Pr[\mathcal{M}(D)\in S]}{\Pr[\mathcal{M}(D')\in S]}$ through Rényi divergence. Wang et al. proposed its privacy amplification lemma [5] when training happens on a random subsample of the entire dataset.

6.4 Preliminaries

In this chapter, we use bold-face letters (e.g. $\mathbf{w}$) to represent vectors, and a subscript to indicate index. We use upper case letters (e.g. W) to represent matrix, except for Q, which represents a query-based dataset (defined later), and q, which represents an individual query in Q. We use $\|\cdot\|$ to denote L_2 norm of a vector or matrix. A summary of symbol notations in this chapter is presented in Table 6.1.

6.4.1 Ranking Problem and Query based data

One key characteristic of ranking model comes from its data type. The training data of a ranking model is (or pre-processed from) a so-called "query"-based data, where $Q = \{q^{(1)}, q^{(2)}, ..., q^{(m)}\}$, are constituted by m queries (also called sessions, or lists), each belongs to one user. A query $q^{(j)}$ is consisted of a series

Table 6.1: Summary of symbol definitions

Symbol	Definition	Symbol	Definition
Q	dataset of queries	f	score function
q	a query based data	s	score output
m	number of queries in a dataset	π	a permutation of documents
j	iterate through queries in a dataset	Ω	space of all permutations
n	number of documents in a query	k	size of prefix of a permutation
i	iterate through documents in a query	α	order of Rényi divergence
$\mathbf{x}$	vector representation of a document	ϵ	privacy parameters
$\mathcal{X}$	domain of document	δ	privacy parameter of approximate DP
y	evaluation on a document	γ	noisy vector for privacy protection
$\mathbf{y}$	evaluation on all documents in a query	G	accumulating gradient
θ	model parameters	t	iteration number
Θ	model space	η	learning rate

of *documents* presented to a user, along with the user's *response* to each document. Formally, let $q^{(j)}$ be a query, then $q^{(j)} = \{\langle \mathbf{x}_i^{(j)}, y_i^{(j)} \rangle\}_{i=1}^{n_j}$, where $\mathbf{x}_i^{(j)} \in \mathcal{X}$ is a vector representation of a document, $y_i^{(j)} \in \mathbb{R}$ is the user's response to that document, for $i = 1, 2, ..., n_j$, and n_j is the number of documents presented to and rated by the user. Let $\mathbf{y}^{(j)} = \langle y_1^{(j)}, \cdots, y_{n_j}^{(j)} \rangle$ denote the vector of truth scores of the documents by user j. (Each user might be presented by a different number of documents, and one document might be presented to multiple users in the dataset Q.)

The goal of training a ranking model is to obtain a scoring function f to evaluate and sort individual documents, which can map a document $\mathbf{x}$ to a score $s \in \mathbb{R}$. If the ranking model is a parametric, such as a neural network, let Θ be the model space, then $f : \mathcal{X}, \Theta \to \mathbb{R}$. In the supervised learning phase, the scores giving by the scoring function by feeding multiple candidate documents of one data point are used to calculate the objective or *loss* of that training example. To use the ranking model in production, a series of candidate documents were first fed into the scoring function, and they are ordered and fed to the user according to the scores giving by the ranking model.

Another characteristic of ranking model comes from its performance evaluation. Different from other machine learning models, which emphasis on optimizing accuracy or objective value on a testing data, for a ranking models, the correctness of relative *order* of the documents in a session plays a significant role. The raw values giving by the scoring function do not matter that much: they are not need to predict

the user's evaluation, only their orders are. Moreover, a model is more preferred if it can perform well on placing the most favored documents into the top positions. (For example, a model which can rank the highest important documents to top 10 positions is more favorable than one who can rank the least important ones to the last positions, since in reality only the top few results are presented to the user.) Due to this property, evaluation criteria such as Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) are often used to evaluate a ranking model.

The calculation of NDCG is supported by most programming languages. By definition, to calculate the NDCG score of one permutation (ranking) π, given the ground truth $\mathbf{y}$ as a vector of true scores of each document, it is calculated by

$$NDCG@k = \frac{DCG@k}{\max DCG@k}, \tag{6.1}$$

and the Discounted Cumulative Gain (DCG) is calculated based on the ground truth scores of the top k documents (according to π):

$$DCG@k := \sum_{i=1}^{k} \frac{2^{\mathbf{y}_{\pi(i)}} - 1}{\log(i+1)}$$

where $\mathbf{y}_{\pi(i)}$ denotes the true score of the i-th ranked document according to the given permutation. And $\max DCG@k$ denotes the DCG calculated according to the perfect ranking, which is the maximum possible value of the document set:

$$\max DCG@k := \sum_{i=1}^{k} \frac{2^{\mathbf{y}_{(i)}} - 1}{\log(i+1)}$$

where $\mathbf{y}_{(i)}$ denotes the i^{th} largest score of $\mathbf{y}$.

Intuitively, NDCG evaluates the quality of the permutation π, with an emphasis on the top positions. MAP is suitable for evaluating ranking performance on binary labels. Suppose labels $l_i \in \{0, 1\}$ for non-Click and Click, then

$$MAP@k := \frac{1}{|\{i|i < k, l_i = 1\}|} \sum_{i=1}^{k} (\text{Precision}@i | l_i = 1) \tag{6.2}$$

For example, if the ranking result is [1, 0, 1, 1, 0, ...], then $MAP@5 = \frac{1}{3}(\frac{1}{1} + \frac{2}{3} + \frac{3}{4}) \approx 0.806$

6.4.2 Listwise Ranking Methods

Listwise ranking methods is a type of ranking models whose objective function is defined based upon training sessions $q^{(i)}$, without pre-processing the sessions into pairs or individual documents. For example, ListNet [97] algorithm defines a probability distribution of permutations, and then define the loss function as the cross entropy loss of two probability distributions.

Definition 6.1 (permutation probability)**.** *For a list of scores* $\mathbf{s}$*, the permutation probability on a permutation* π *is:*

$$P_{\mathbf{s}}(\pi) = \prod_{j=1}^{n} \frac{\psi(s_{\pi(j)})}{\sum_{k=j}^{n} \psi(s_{\pi(k)})} \tag{6.3}$$

where ψ *is a transformation of the score (e.g.* $\psi(s) = \exp(s)$ *are often used), and* $\pi(j)$ *is the* j*-th item of* π*.*

It can be shown that the permutation probabilities $P_{\mathbf{s}}(\pi)$ forms a probability distribution, and the probability is higher if the permutation π is more in accordance with the orders of the scores $\mathbf{s}$.

Since the number of permutations $O(n!)$ is intractable, we can take top k probability of documents $\langle j_1, j_2, ..., j_k \rangle$, representing the probability of their being ranked on the top k positions, given the score of all the documents.

$$P_{\mathbf{s}}(\langle j_1, j_2, ..., j_k \rangle) = \sum_{\pi \in \langle j_1, j_2, ..., j_k \rangle} P_{\mathbf{s}}(\pi) = \prod_{t=1}^{k} \frac{\psi(s_{j_t})}{\sum_{l=t}^{n} \psi(s_{j_l})} \tag{6.4}$$

where $P_{\mathbf{s}}(\pi)$ is the permutation probability of π given score list s, and s_{j_t} is the score of object j_t which is ranked in position t, $t = 1, ..., n$. Therefore, in order to calculate these $\frac{n!}{(n-k)!}$ top k probabilities, we no longer need to calculate the $n!$ permutation probabilities. And it can be shown that top-k probabilities also form a probability distribution. In practice, we can take $k = 1$ to reduce the computation complexity.

Based on the permutation probability, ListNet defines a surrogate loss function as the cross entropy of the probability permutation given by the truth y, and the probability permutation given by the scores:

$$R(f(x), y) = CE[P(\pi|\mathbf{x}; y) \| P(\pi|\mathbf{x}; f)] = -\sum_{\forall \pi \in \Omega} P(\pi|\mathbf{x}; y) \log P(\pi|\mathbf{x}; f)$$

where $P(\pi|\mathbf{x}; y)$ is the permutation (or top k) probability according to truth (y); and $P(\pi|\mathbf{x}; f)$ is the permutation (or top k) probability according to the scores. Given two lists of scores, we can first calculate

the two corresponding permutation probability distributions, and then take the metric between the two distributions as the listwise loss function.

BayesRank [103] is a ranking method which aims to *directly* optimize the performance of IR applications in terms of various evaluation measures (such as MAP or NDCG). Similar as the pairwise method LambdaRank [94], it involves calculation of NDCG during the training process, but its data consumption is list-wise instead of pair-wise, like ListNet.

BayesRank consider ranking as a decision making problem and evaluate the *Bayes risk* of the ground truth: Let $\ell(\pi; \pi_q^*)$ be the permutation loss incurred by taking a ranking decision when perfect ranking is $\pi_q^* = sort_y\{d_1, ..., d_n\}$ (the documents are sorted according to the ground truth y), given a query q. In the retrieval stage, no explicit information about π_q^* is presented. This means, any arbitrary permutation could be π_q^*. Therefore, we model the uncertainty by the conditional probability $p(\pi; q)$, which corresponds to the probability that π would be judged as the perfect permutation for query q. From the Bayesian decision theory, the expected risk of taking the decision is given by

$$R(\pi; q) = \int_{\pi'} \ell(\pi, \pi') \, \mathrm{d}p(\pi'; q) \, .$$

The best decision can be selected by minimizing the expected risk as $R(\pi; q)$, which is called the Bayes risk. In supervised learning phase, the ground-truth associated with each query q is presented. Hence, the Bayes risk of π_q^* over the training query set Q is

$$R(\pi^*; Q) = \sum_{q \in Q} p(q) R(\pi_q^*; q) = \sum_q p(q) \int_{\pi'} \ell(\pi, \pi') \, \mathrm{d}p(\pi'; q) \, ,$$

if we assume Q contains i.i.d. samples. Since permutation space Π_q is finite, the expected Bayes risk an be approximated by

$$R \approx \frac{1}{m} \sum_{q \in Q} \sum_{\pi' \in \Pi_q} \ell(\pi_q^*, \pi') p(\pi'; q) \tag{6.5}$$

where the scoring function is embedded in the conditional probability $p(\pi'; q)$. In this way, it no longer formulate the ranking error as a surrogate loss function; instead, the permutation-level loss ℓ can be directly related to the desired IR evaluation measures, such as MAP and NDCG.

One key feature of this method is that, the ground truth of training session (i.e. user's preference) is only involved in calculating the IR evaluation measures.

Another listwise training model, ListMLE [99], defined the surrogate loss function as:

$$R(f, \langle \mathbf{x}_i, y_i \rangle_{i=1}^{n_j}) = -\log P(\langle y_i \rangle_{i=1}^{n_j} | \langle \mathbf{x} \rangle_{i=1}^{n_j}; f) = -\log \prod_{i=1}^{n} \frac{\exp(f(\mathbf{x}_{y(i)}))}{\sum_{k=i}^{n} \exp(f(\mathbf{x}_{y(k)}))}, \tag{6.6}$$

where $\mathbf{x}_{y_{(i)}}$ is the i-th ranked document according to the ground truth.

6.5 Algorithms

This section first introduces our definition of a new privacy framework, fitting the properties of the session-wise training data, and then describes each component of the proposed algorithm to achieve that framework.

6.5.1 User-preference level privacy

For the session-wise privacy, each individual is represented by a session historically exposed to a user. The goal should be protecting user's preferences on the documents, i.e. $\{y_i^{(j)}\}_{i=1}^{n_j}$ for each user j. To be specific, given a dataset of queries Q, each session $q^{(j)} = \{\langle \mathbf{x}_1^{(j)}, y_1^{(j)} \rangle, \langle \mathbf{x}_2^{(j)}, y_2^{(j)} \rangle, ..., \langle \mathbf{x}_{n_j}^{(j)}, y_{n_j}^{(j)} \rangle\}$ consists a set of documents $\{\mathbf{x}_i^{(j)}\}_{i=1}^{n_j}$ and a set of user's judgements (evaluations) $\{y_i^{(j)}\}_{i=1}^{n_j}$. We argue that, in many applications, an reasonable privacy protection goal is to protect the user's evaluations, for the following reasons. (i) In most of the applications, for a session based data, only the evaluation part comes from the users, and some applications may be interested in only protecting data collected from users. For example, user's response (truth scores) to online products represents, user's click and reading of documents, etc. (ii) The documents (i.e. feature vectors $\{\mathbf{x}_i^{(j)}\}$) are generated and presented to the user before the user makes a choice among them, therefore, they are independent from $\{y_i^{(j)}\}$. (iii) The documents are not independent from session-to-session: multiple users in Q might be exposed to the same document. So it is hard to define the effect of Q in case of presence or absence of one document.

Therefore, we can define a type of session-wise privacy considering only the user data as sensitive information, given by our definition of neighboring datasets as below:

Definition 6.2 (User level neighboring and privacy)**.** *Define datasets $Q \overset{q}{\sim} Q'$ if Q and Q' are differed by the user's preference (scores) of one session. To be specific, if $Q = \{q^{(1)}, q^{(2)}, ..., q^{(m)}\}$ and $Q' = \{q'^{(1)}, q'^{(2)}, ..., q'^{(m)}\}$ are session-based data, then $Q \overset{q}{\sim} Q'$ if there exists one index $j \in [m]$ where $q^{(j)} =$*

$\langle \mathbf{x}^{(j)}, y^{(j)} \rangle$ *and* $q'^{(j)} = \langle \mathbf{x}'^{(j)}, y'^{(j)} \rangle$*, such that* $\mathbf{x}^{(j)} = \mathbf{x}'^{(j)}$*,* $y^{(j)} \neq y'^{(j)}$*, and* $q^{(k)} = q'^{(k)}$ *for* $k \in [m]$ *and* $k \neq j$*.*

A randomized mechanism $\mathcal{M}$ *is session-wise* (ϵ, δ)*-DP, for privacy parameters* $\epsilon \geq 0, 0 \leq \delta < 1$*, if for all session-wise neighboring datasets* $Q \overset{q}{\sim} Q'$ *and all events* $S \subseteq range(\mathcal{M})$*,*

$$\Pr[\mathcal{M}(Q) \in S] \leq \exp(\epsilon) \Pr[\mathcal{M}(Q') \in S] + \delta$$

This framework applies on protecting the strong adversity with arbitrary side information who knows all but one user's preference on the series of provided documents: if the training mechanism satisfies session-wise DP, his ability on gaining confidence on inferring the target user's preference is limited.

6.5.2 Private BayesRank Algorithm

Our private algorithm are based on the Listwise approaches introduced in Section 6.4.2, especially the BayesRank algorithm. To train the BayesRank model through stochastic gradient descent, we need to factorize the loss function and evaluate the gradient of each part. First, the permutation-level loss in (6.5) would naturally be the negative of NDCG score:

$$\ell(\pi_q^*, \pi) = -NDCG@k(\pi_q^*, \pi)$$

The choice of NDCG has these benefits comparing to MAP: (i) NDCG allows for numeric scores, while MAP applies for binary labels; (ii) when k is chosen to be a small integer (like $k = 1$), MAP@k might not be effectively calculated. For a given permutation π, we can divide the document list into two sub-lists: π_1^k and $\pi_{k+1}^{n_j}$, as the first sub-list affects NDCG score, while the second sub-list does not.

$$\pi = \langle \pi_{(1)}, ..., \pi_{(k)}, \pi_{(k+1)}, ..., \pi_{(n_j)} \rangle = \langle \pi_1^k, \pi_{k+1}^{n_j} \rangle$$

Therefore, we can evaluate the loss by only considering the top k documents of the permutation π. So, the loss function can be re-written as

$$R(Q, \theta) = -\frac{1}{m} \sum_{q_j \in Q} \sum_{\pi_1^k} NDCG@k(\mathbf{y}^{(j)}, \pi_1^k) p_k(\pi_1^k; \mathbf{x}^{(j)}, \theta) \tag{6.7}$$

where θ is the set of parameters in the neural networks. Here, according to (6.4), the permutation probability is

$$p_k(\pi_1^k; \mathbf{x}^{(j)}, \theta) = \prod_{i=1}^{k} \frac{\exp(f(\mathbf{x}_i^{(j)}, \theta))}{\sum_{l=i}^{k} \exp(f(\mathbf{x}_l^{(j)})) + \sum_{\mathbf{x}_l \notin \pi_1^k} \exp(f(\mathbf{x}_l^{(j)}))},$$

when we choose exp as the transformation function ψ.

In practice, we usually choose $k = 1$ for efficiency. The privatised algorithm we proposed achieves differential privacy through gradient perturbation, with Gaussian noise injected into the sensitive-data dependent intermediate results: NDCG calculation. The pseudo-code is presented in algorithm 6.1.

Algorithm 6.1 Private BayesRank (PRIVBR)

Input: training sessions $Q = \langle X, Y \rangle$, where $X = \{\mathbf{x}^{(j)}\}$ for $j = 1, ..., m$ is the set of document lists fed to m users: $\mathbf{x}^{(j)} = \{\mathbf{x}_i^{(j)}\}$, for $i = 1, ..., n_j$, each be a feature vector. $Y = \{\mathbf{y}^{(j)}\}$ is the set of user's preferences collected from them, $\mathbf{y}^{(j)} = \{y_i^{(j)}\}$, for $i = 1, ..., n_j$, each be a user's preference value for $x_i^{(j)}$. Y is the private data which needs protection. Hyperparameter: k, sampling ratio q, step size η_t
Initialize neural network model f, model parameters θ_0
for $t = 0, 1, ...$ **do**
    Sample a mini-batch $B \subseteq [m]$ by sampling ratio q.
    for $j \in B$ **do**
        $G_\theta^{(j)} \leftarrow 0$
        for $i = 1, ..., n_j$ **do**
            Evaluate scores $\exp(f(\mathbf{x}_i^{(j)}, \theta))$ and score gradients $\frac{\partial f(\mathbf{x}_i^{(j)}, \theta)}{\partial \theta}$ ▷ Pre-calculation
        end for
        for Each top k permutation π_1^k **do**
            Evaluate $NDCG@k(\pi_1^k, \mathbf{y}^{(j)})$
            for $i = 1, ..., n_j$ **do**
                Evaluate $\frac{\partial p_k(\pi_1^k; \mathbf{x}^{(j)}, \theta)}{\partial f(\mathbf{x}_i^{(j)}, \theta)}$ and $\frac{\partial f(\mathbf{x}_i^{(j)}, \theta)}{\partial \theta}$
                $G_\theta^{(j)} \leftarrow G_\theta^{(j)} + NDCG@k(\pi_1^k, \mathbf{y}^{(j)}) \times \frac{\partial p_k(\pi_1^k; \mathbf{x}^{(j)}, \theta)}{\partial f(\mathbf{x}_i^{(j)}, \theta)} \times \frac{\partial f(\mathbf{x}_i^{(j)}, \theta)}{\partial \theta}$ ▷ Accumulate gradients
            end for
        end for
    end for
    $G_\theta \leftarrow \sum_{j \in B} G_\theta^{(j)}$
    Sample noise vector γ ▷ See 6.5.3
    Privatize $G_\theta \leftarrow G_\theta + \gamma$
    Update model $\theta_{t+1} \leftarrow \theta_t - \eta_t G_\theta$
end for

6.5.3 Sensitivity calculation

According to our definition of user-preference level privacy, the privacy leak incurs in the calculation of NDCG (Line 11), since it requires accessing the user's preference, $\mathbf{y}^{(j)}$, and NDCG is bounded in $[0, 1]$. The final gradient calculated on one list, $G_\theta^{(j)}$, is a sum of NDCG multiply by two *factored* gradients, where $\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)}$ is the partial gradient of permutation probability with respect to the score, and $\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial \theta}$ is the partial gradient of score with respect to the model weights. Now we argue that *both* of these terms are independent of the sensitive data, $\mathbf{y}^{(j)}$.

Partial gradient of permutation probability with respect to the score

The function for calculating $\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)}$ is:

$$\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)} = p_k(\pi_1^k;\mathbf{x}^{(j)},\theta) \times \begin{cases} 1 - \sum_{i=1}^{rank(d)} \frac{\exp(f(\mathbf{x}_d^{(j)},\theta))}{\sum_{l=i}^{k}\exp(f(\mathbf{x}_l^{(j)},\theta)) + \sum_{\mathbf{x}_l \notin \pi_1^k}\exp(f(\mathbf{x}_l^{(j)},\theta))} & \text{if } d \in \pi_1^k \\ - \sum_{i=1}^{k} \frac{\exp(f(\mathbf{x}_d^{(j)},\theta))}{\sum_{l=i}^{k}\exp(f(\mathbf{x}_l^{(j)},\theta)) + \sum_{\mathbf{x}_l \notin \pi_1^k}\exp(f(\mathbf{x}_l^{(j)},\theta))} & \text{if } d \notin \pi_1^k \end{cases}$$

and $rank(d)$ denotes the rank of document d in the currently processing prefix list π_1^k, which does not depends on the true value (or order) of the document d. The permutation probability $p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)$ defined in (6.3) only depends the scores predicted by the current model f, so this term only depends on model vector θ and each document feature, $\mathbf{x}$.

Partial gradient of score with respect to model

The second part of multipliers, $\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial \theta}$, is related to the structure of the scoring model. We will show two examples of parametric models: (i) linear scoring function: $f(\mathbf{x},\theta) = \theta \cdot \mathbf{x}$; (ii) multi-layer perception (MLP) model. Both models are widely used scoring functions for ranking.

(i) For the simplest situation of linear scoring function, to calculate the score, $s = f(\mathbf{x},\theta) = \mathbf{x} \cdot \theta$, which does not involve Y; to calculate the gradient, $\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial \theta} = \mathbf{x}$, which does not involve Y, either.

(ii) For a more complex neural network model, assume there is one hidden layer of width m. (Situation is similar if there are more than one hidden layers). Let the model parameters be $\theta = \{W_1, \mathbf{b}_2, W_2, b_2\}$. ($b_2$ is not needed for ranking models, but we still include it for completion.)

Forward phase

$$\begin{aligned}
\mathbf{h}_1 &= W_1\mathbf{x} + \mathbf{b}_1 && \mathbf{x} \in \mathbb{R}^n, W_1 \in \mathbb{R}^{m\times n}, \mathbf{b}_1 \in \mathbb{R}^m \\
\mathbf{a}_1 &= \phi(\mathbf{h}_1) && \text{element-wise} \\
h_2 &= W_2\mathbf{a}_1 + b_2 && W_2 \in \mathbb{R}^{m\times 1}, b_2 \in \mathbb{R}
\end{aligned}$$

The output of neural network is $h_2 := f(\mathbf{x}, \theta)$. The output is also called the *score* of the input $\mathbf{x}$, which is used for ranking in testing phase.

Backward phase, the NDCG, $\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)}$, and $\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial \theta}$ are calculated separately. To calculate $\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial \theta}$ by back-propagation,

$$\begin{aligned}
\frac{\partial}{\partial W_2} f(\mathbf{x}, \theta) &= \frac{\partial}{\partial W_2} W_2\mathbf{a}_1 + b_2 = \mathbf{a}_1 \\
\frac{\partial}{\partial b_2} f(\mathbf{x}, \theta) &= \frac{\partial}{\partial b_2} W_2\mathbf{a}_1 + b_2 = 1 \\
\frac{\partial}{\partial W_1} f(\mathbf{x}, \theta) &= \frac{\partial}{\partial W_1} W_2\mathbf{a}_1 + b_2 = \frac{\partial}{\partial W_1} W_2\phi(W_1\mathbf{x} + \mathbf{b}_1) = W_2 \odot \phi'(\mathbf{a}_1)\mathbf{x}^T \\
\frac{\partial}{\partial \mathbf{b}_1} f(\mathbf{x}, \theta) &= \frac{\partial}{\partial b_1} W_2\mathbf{a}_1 + b_2 = \frac{\partial}{\partial \mathbf{b}_1} W_2\phi(W_1\mathbf{x} + \mathbf{b}_1) = W_2 \odot \phi'(\mathbf{a}_1)
\end{aligned}$$

Since $\mathbf{y}$ is not involved in the calculation, this part is also independent of sensitive data.

Scale the sensitivity

To scale the sensitivity of the gradient G_θ, we need to analyze how the two factors of gradients effect on the NDCG. Consider B a mini-batch of the dataset Q, and its neighbor, B', which differs by one list of ground truths $\mathbf{y}^{(*)}$ and $(\mathbf{y}')^{(*)}$ at some point j. We would like to see the difference between two gradients G_θ and G'_θ, calculated on B and B', respectively.

$$
\begin{aligned}
&\|G_\theta - G'_\theta\| \\
=&\|\sum_{j\in B}\sum_{\pi_1^k}\sum_{i=1}^{n_j}\Bigg[NDCG@k(\pi_1^k,\mathbf{y}^{(j)})\times\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)}\times\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial\theta}- \\
&NDCG@k(\pi_1^k,(\mathbf{y}')^{(j)})\times\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)}\times\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial\theta}\Bigg]\| \\
=&\|\sum_{j\in B}\sum_{\pi_1^k}\left|NDCG@k(\pi_1^k,\mathbf{y}^{(j)})-NDCG@k(\pi_1^k,(\mathbf{y}')^{(j)})\right|\sum_{i=1}^{n_j}\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)}\times\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial\theta}\| \\
\leq&\max_{j\in B}\sum_{\pi_1^k}\left|NDCG@k(\pi_1^k,\mathbf{y}^{(j)})-NDCG@k(\pi_1^k,(\mathbf{y}')^{(j)})\right|\|\sum_{i=1}^{n_j}\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)}\times\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial\theta}\| \\
\leq&\max_{j\in B}\sum_{\pi_1^k}\|\sum_{i=1}^{n_j}\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)}\times\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial\theta}\|
\end{aligned}
$$

where $\|\cdot\|$ denotes the L_2 norm. The first inequality holds due to that Q and Q' only differs by the scores of one session. The second inequality holds due to that NDCG is bounded by $[0,1]$, so $|NDCG@k(\pi_1^k,\mathbf{y}^{(j)})-NDCG@k(\pi_1^k,(\mathbf{y}')^{(j)})|\leq 1$.

The sensitivity $\|G_w - G'_w\|$ can be monitored during training: when the gradient with respect to the model is calculated, at each iteration of a list in B, we can trace the term $\sum_{\pi_1^k}\|\sum_{i=1}^{n_j}\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)}\times\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial\theta}\|$ for each list and document being processed, and store the maximum over all list in the current mini-batch.

Let C be the maximum value over all the lists in the training mini-batch, then we can inject noise $\gamma\sim\mathcal{N}(0,\sigma^2C^2)$ into the accumulated gradient G_θ to achieve differential privacy.

6.5.4 Integrating with Spectral Normalization

For complex models such as neural network, if there is no restrictions on the model parameters, the calculated sensitivity, $\|\sum_{i=1}^{n_j}\frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)}\times\frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial\theta}\|$, could be unbounded, resulting a large noise injection. Therefore, we can combine this algorithm with some regularization strategy. Although existing private neural networks [14] incorporate gradient clipping to control the noise injection, it has a drawback

that an apriori clipping threshold needs to be chosen, and it is hard to adaptively change it during the training phase, so when the SGD pushes the model to a local optimal, the threshold might be too large.

Therefore, we choose the spectral normalization [105] technique for deep neural network to control the norm of gradient of neural network, $\frac{\partial f(\mathbf{x}^{(j)},\theta)}{\partial \theta}$. This technique was introduced as a regularization technique in training discriminator of GAN in non-private settings. After each update of weight matrices $W_1, W_2, ...$, the spectral normalization technique normalizes the spectral norm of the weight matrix so that it satisfies the Lipschitz constraint $\sigma(W) = 1$:

$$W \leftarrow W/\sigma(W)$$

where $\sigma(A)$ is the spectral norm (largest singular value) of the matrix A. In our experiments, we find that spectral normalization helps bounding the sensitivity since it regularizes the norm of the model weights, especially when the the model is updated by gradients with noisy perturbation.

6.5.5 Further reduce the sensitivity

Although the NDCG is upper-bounded by 1, in most real datasets, the NDCGs of most Top-k permutations are not as large as 1, and would even reduce to 0 if all documents in π_1^k of that permutation have a score of 0. For the case of $k = 1$, NDCG is zero as long as the the score of the document used to calculate NDCG@1 is zero. For example, if a session q has truth values $\mathbf{y} = \{0, \cdots, 0, 1, 0, \cdots, 0\}$ (i.e. only one document in the session has a non-negative score). In such case, one of the n_j Top-1 permutations has NDCG of 1, all the permutations have NDCG 0. From this observation, we can conclude that if the session length is n_j but there are at most p documents having a non-zero evaluation score, then at most p of the n_j top-1 permutations have a non-zero NDCG.

In many ranking tasks, most training sessions has only one to two non-zero scores. For example, in the click-through dataset, where the event is that the user clicked the document, since most users would not click too many documents presented to them, usually one session would only contain very few positive events.

If we assume all possible session has only one non-zero score (as an assumption), then the calculated sensitivity $\sum_{\pi_1^k} \| \sum_{i=1}^{n_j} \frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)} \times \frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial \theta} \|$ can be reduced to $\max_{\pi_1,\pi_2} \sum_{\pi_1,\pi_2} \| \sum_{i=1}^{n_j} \frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)} \times \frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial \theta} \|$ (i.e. the sum of the two largest values of $\| \sum_{i=1}^{n_j} \frac{\partial p_k(\pi_1^k;\mathbf{x}^{(j)},\theta)}{\partial f(\mathbf{x}_i^{(j)},\theta)} \times \frac{\partial f(\mathbf{x}_i^{(j)},\theta)}{\partial \theta} \|$ over all top-1 permu-

tations). This is because a neighboring session would only have one positive evaluations, so at most two Top-1 permutations would have positive NDCG. However, this is a strong assumption which does not hold for most of the datasets: even if the chance to have a session with multiple positive events is extremely low, since it is part of the user's preference, those cases needs to be protected to meet the privacy guarantee.

In stead, we can let the algorithm to pre-process each training sample during training, by constricting the number of non-zero scores: if the training sample has more than one non-zero elements, it pre-process the data to randomly select and keep only one of the highest non-zero scores. This pre-process would guarantee that the neighboring datasets only differ by at most two of the $|\pi_1^k|$ permutations, thus further reducing the sensitivity. Although this would loss some training signals in the gradient, our experiments shows that decrease in the noise injection would overweight the information loss.

6.6 Experimental Results

6.6.1 Datasets

We tested the ranking performance of all private ranking algorithms on the two LETOR 4.0 datasets [106]: MQ2007 and MQ2008. LETOR 4.0 is a package of benchmark datasets for research on Learning to Rank. Both datasets are webpage retrieval datasets, which use the Gov2 web page collection and two query sets from Million Query track of TREC2007 and TREC 2008, respectively. We use the supervised ranking part of the datasets, which is supported with relevance labels. All the attributes of the documents are anonymized. From the data description, they contain information retrieval features such as TF, IDF, DL, BM25, LMIR, etc.

For MQ2007, data size $m = 1,692$, feature length $p = 46$, and session length n_j ranges from 6 to 147, with majority $(1549/1692)$ of them having a length of 40; For MQ2008, $m = 784$, $p = 46$, and session length n_j ranges from 5 to 121, with majority $(339/784)$ of them having a length of 8. Both datasets have response values ranging from 0.0 to 2.0. Since list-wise ranking models consider each session as a training individuals, the training data are not pre-processed: each feature vector is directly fed into the scoring model. For model evaluation, all positive evaluations are considered as 1.0 when calculating MAP.

6.6.2 Model and baselines

As suggested in the non-private algorithms [103], we perform our experiment on a multi-layer perceptron (MLP) model with one hidden layer. Other than the BayesRank model, we tested two other popular listwise models: ListNet [97], which is is cross-entropy between the permutation probability of the ground truth and the permutation probability of the scores, and ListMLE [99], which applies a consistent and sound surrogate loss function.

To privatise these two algorithms, we incorporate the gradient clipping method presented in [14], i.e. clip and perturb the sum of mini-batch gradient before SGD update. We also apply the gradient clipping method on BayesRank as a baseline algorithm. We apply 5-fold cross validation on all algorithms, by the train-test split provided by the data provider. Due to the randomness of the private algorithm, each fold is repeated 10 times, and the average performances on the testing data are reported.

6.6.3 Hyperparameter setting

We set $\delta = 10^{-4}$ in our experiments, which is a value between $1/n$ and $1/n^2$. For the MLP model, we add a bias threshold to the input layer, and one hidden layer with width 256. No bias threshold is added into the hidden layer, since it is not necessary, and the output layer as only one neuron, which indicate the score of the input.

Four different noisy scales were tested on each dataset, ranging from high noise (low ϵ) to low noise (high ϵ). The mini-batch size was fixed to 10, which can largely benefit from privacy amplification through sub-sampling effect. In this way, for each group of experiments, privacy ϵ accumulates at the same speed for all private algorithms. Although non-private algorithms would converge in about 100 iterations, each experiment is trained for 400 iterations, to allow enough training for convergence for the private algorithms. After tuning on the non-private algorithm, we set a initial learning rate η_0 of 0.1 for BayesRank and ListNet algorithms, and 0.01 for ListMLE algorithms, and a linear decrease every 100 iterations.

6.6.4 Results

Note that since we use session-wise algorithms, both datasets are not large in consider of sizes: MQ2007 contains $1,692$ users, and MQ2008 contains 784 users. It is a known fact that protecting the privacy of a

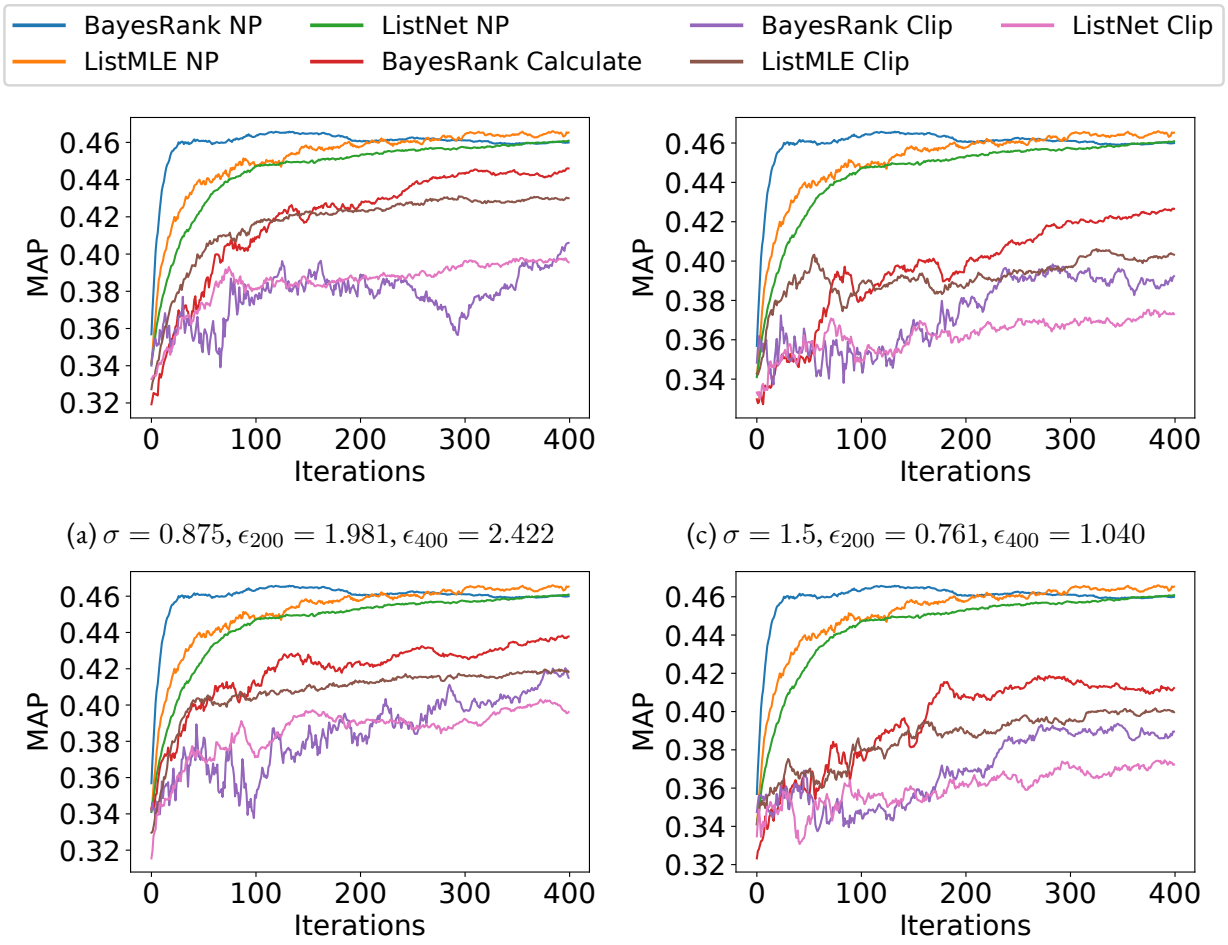
small dataset is more difficult than doing so in a large dataset, due to the nature of "averaging" of many machine learning models, the effect of one individual get stronger dilution in a larger dataset.

Figure 6.1 and 6.2 show the performance of the private algorithms on MQ2007 and MQ2008 datasets, respectively. For each figure, we show the evaluation criteria against training iterations, for criteria of MAP, NDCG@1, NDCG@2, NDCG@4, and NDCG@8, from top to bottom, on the testing dataset. Performance of non-private (NP) baseline algorithms of BayesRank, ListMLE, and ListNet are also plotted. We choose 4 different noisy perturbation scales σ^2, which result in a range of around $0.75 - 2.5$ of private parameter ϵ after 400 iterations, so that we can see privacy-utility trade-offs at a reasonable privacy level.

From the experimental results, we can see that private algorithms converges at a lower speed compare to non-private algorithms in general, due to the noisy perturbation. Smaller noisy perturbation would result in performance closer to non-private algorithms, but there is still a gap between PrivBR and the non-private baselines, especially on the MQ2007 dataset. In MQ2008 dataset, the experiment with the smallest perturbation (largest ϵ) shows the performance closer to non-private baselines, especially for NDCG@1 and NDCG@2. When achieving high privacy is required (like $\epsilon < 1$), the drop in performance from non-private algorithm in NDCGs is about 0.1 (like for NDCG8, it drops from 0.45 to 0.35). In all experiments, the PrivBR achieves the highest performance comparing to other baselines. This indicates that, if the application allows to consider the feature vector $\mathbf{x}$ as public information, while only requires protecting the user's preference $\mathbf{y}$, applying the sensitivity calculation technique of PrivBR can increase the utility of the model. The private algorithms applying gradient clipping does not require the feature vectors to be public. So if it is needed to protect that part of training data, applying gradient clipping algorithm with BayesRank or ListMLE can still achieve reasonable performance.

6.7 Conclusions

In this chapter, an analysis on achieving differential privacy on training a machine learning ranking model is presented. We consider users who made preference on the provided documents as individuals, therefore listwise ranking techniques are considered. We propose a framework of privacy applicable to listwise training data, and perform a sensitivity analysis on the BayesRank model. Experimental results show that sensitivity calculation technique can result in a higher utility of the private model.



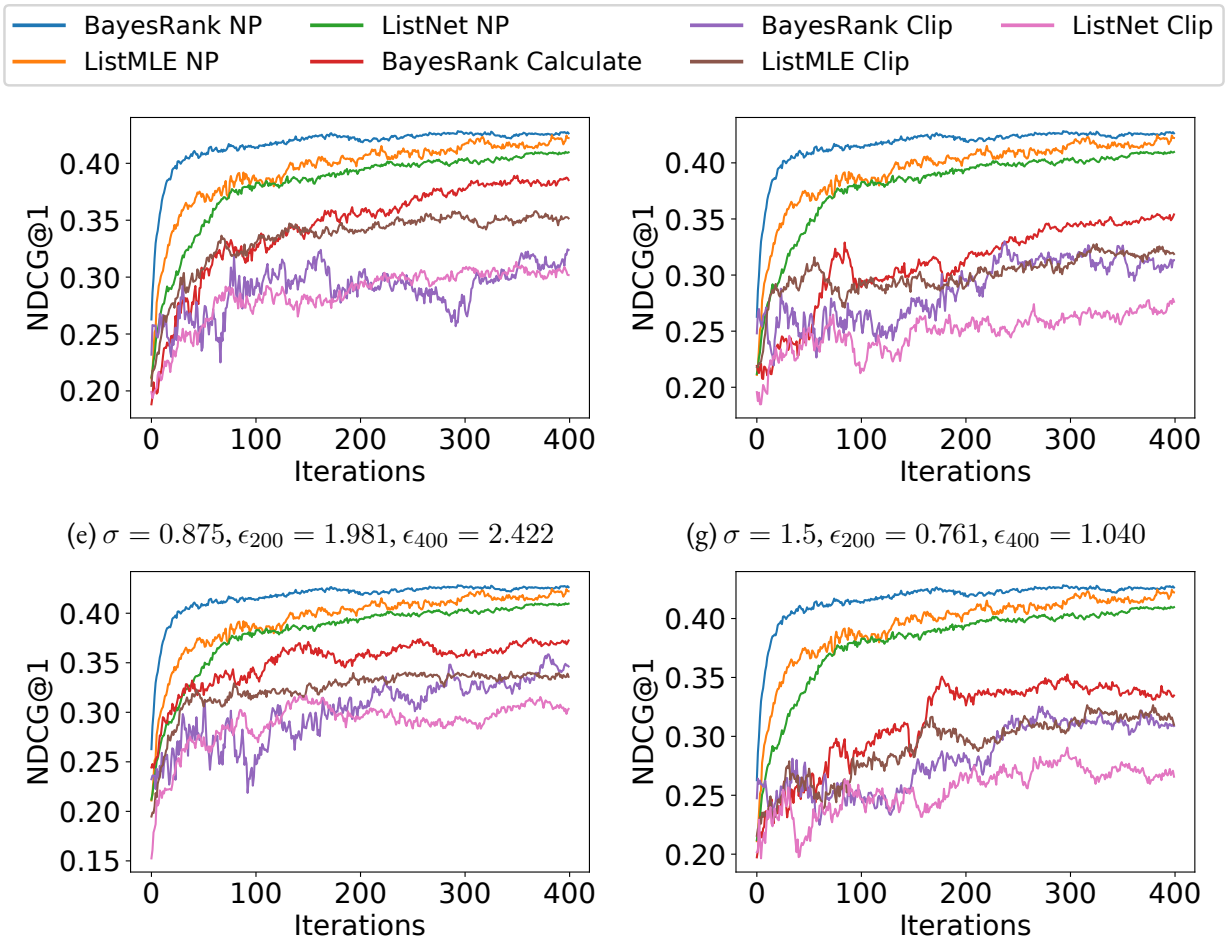
(a) $\sigma = 0.875, \epsilon_{200} = 1.981, \epsilon_{400} = 2.422$

(c) $\sigma = 1.5, \epsilon_{200} = 0.761, \epsilon_{400} = 1.040$

(b) $\sigma = 1.0, \epsilon_{200} = 1.484, \epsilon_{400} = 1.817$

(d) $\sigma = 2.0, \epsilon_{200} = 0.0.551, \epsilon_{400} = 0.758$

Figure 6.1: 1 of 5. Performance on LETOR 4.0 dataset MQ2007, MAP.



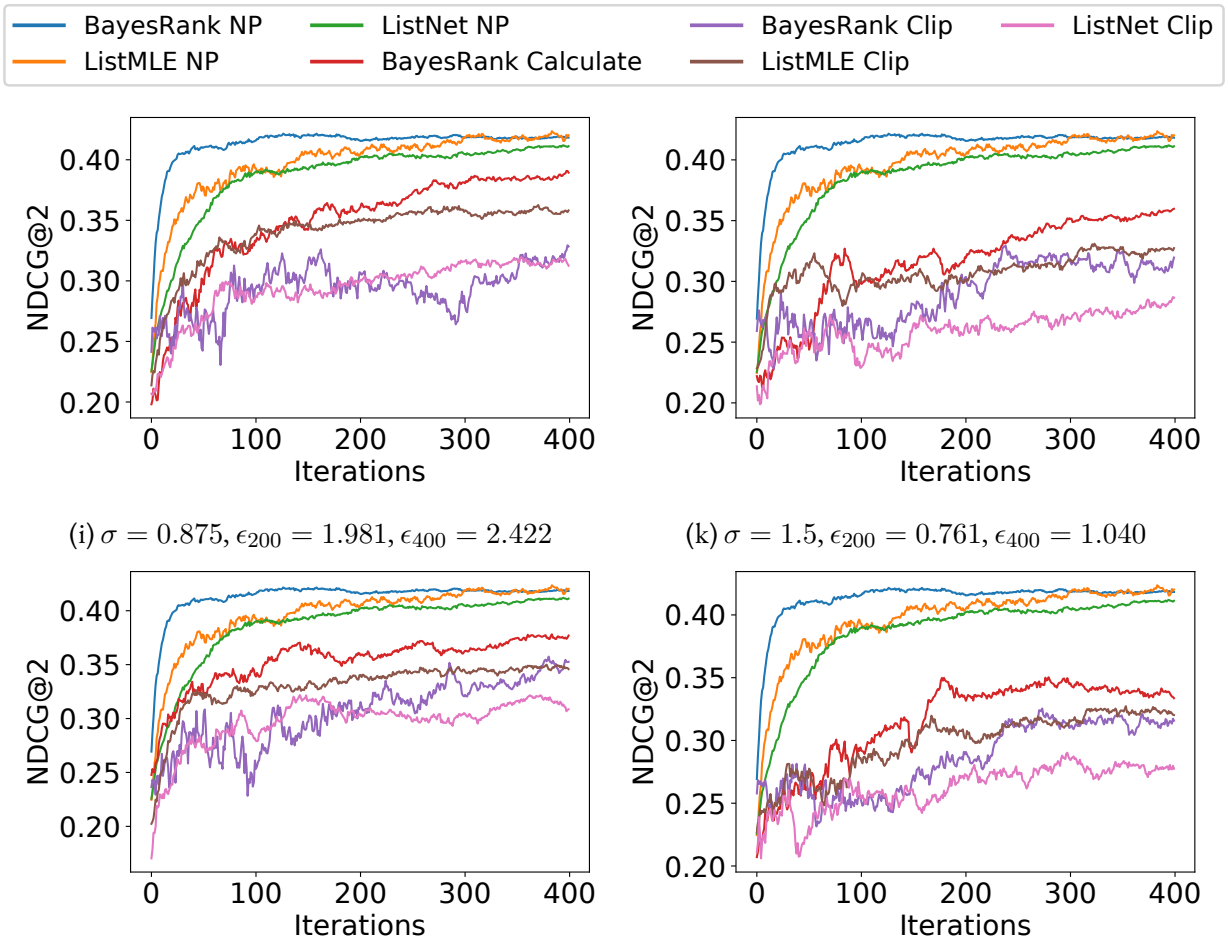
(e) $\sigma = 0.875, \epsilon_{200} = 1.981, \epsilon_{400} = 2.422$

(g) $\sigma = 1.5, \epsilon_{200} = 0.761, \epsilon_{400} = 1.040$

(f) $\sigma = 1.0, \epsilon_{200} = 1.484, \epsilon_{400} = 1.817$

(h) $\sigma = 2.0, \epsilon_{200} = 0.0.551, \epsilon_{400} = 0.758$

Figure 6.1: 2 of 5. Performance on LETOR 4.0 dataset MQ2007, NDCG@1.



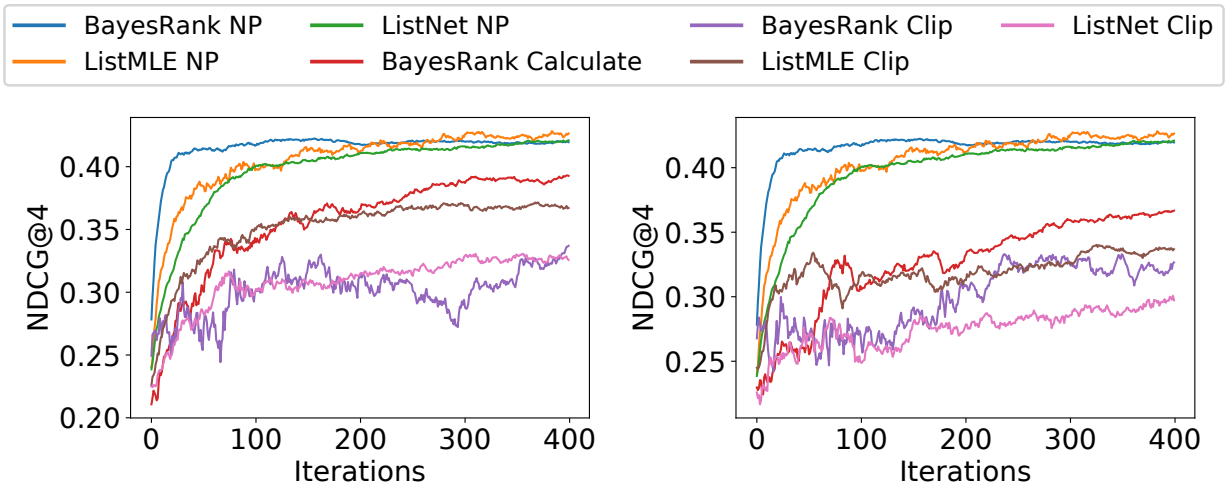
(i) $\sigma = 0.875, \epsilon_{200} = 1.981, \epsilon_{400} = 2.422$

(k) $\sigma = 1.5, \epsilon_{200} = 0.761, \epsilon_{400} = 1.040$

(j) $\sigma = 1.0, \epsilon_{200} = 1.484, \epsilon_{400} = 1.817$

(l) $\sigma = 2.0, \epsilon_{200} = 0.0.551, \epsilon_{400} = 0.758$

Figure 6.1: 3 of 5. Performance on LETOR 4.0 dataset MQ2007, NDCG@2.



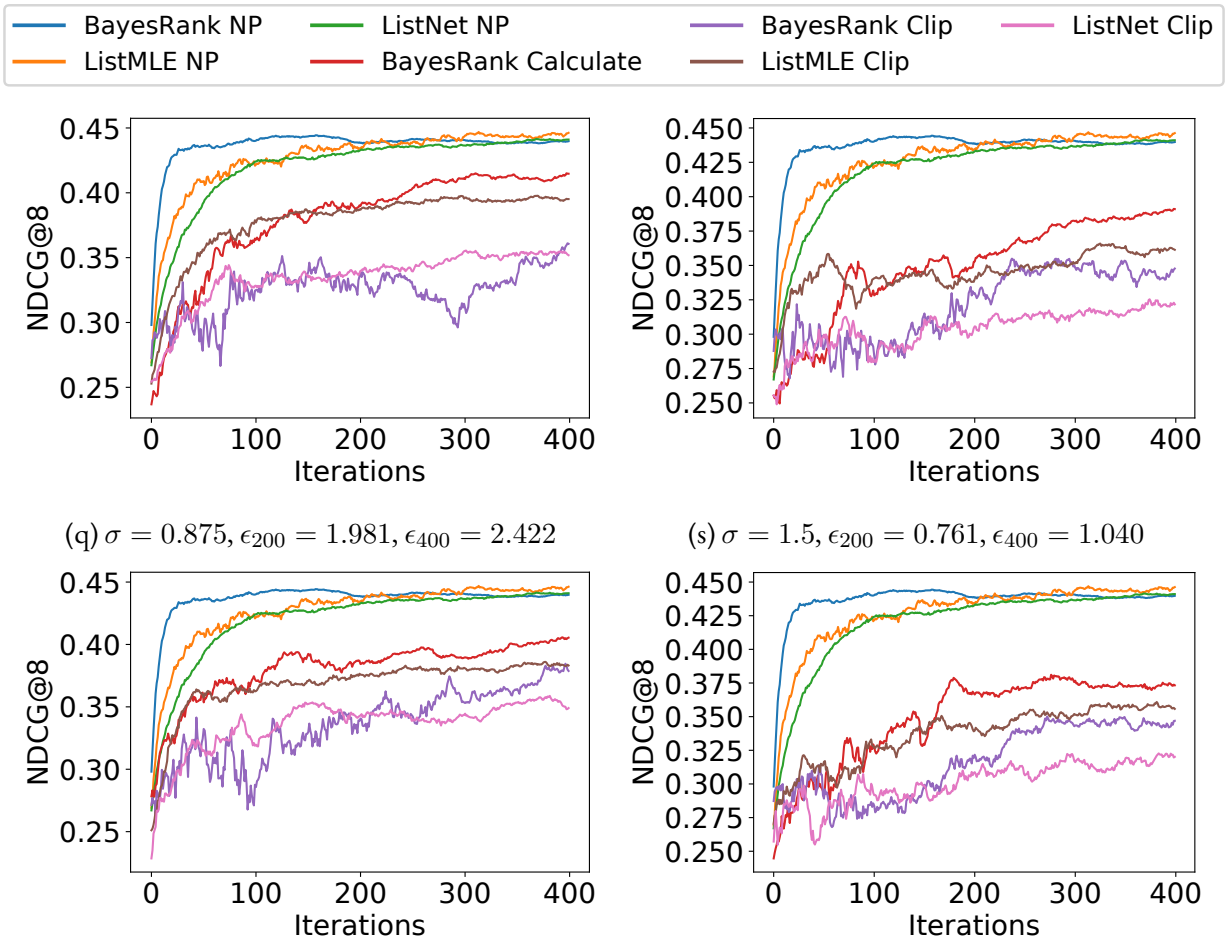
(m) $\sigma = 0.875, \epsilon_{200} = 1.981, \epsilon_{400} = 2.422$

(o) $\sigma = 1.5, \epsilon_{200} = 0.761, \epsilon_{400} = 1.040$

(n) $\sigma = 1.0, \epsilon_{200} = 1.484, \epsilon_{400} = 1.817$

(p) $\sigma = 2.0, \epsilon_{200} = 0.0.551, \epsilon_{400} = 0.758$

Figure 6.1: 4 of 5. Performance on LETOR 4.0 dataset MQ2007, NDCG@4.



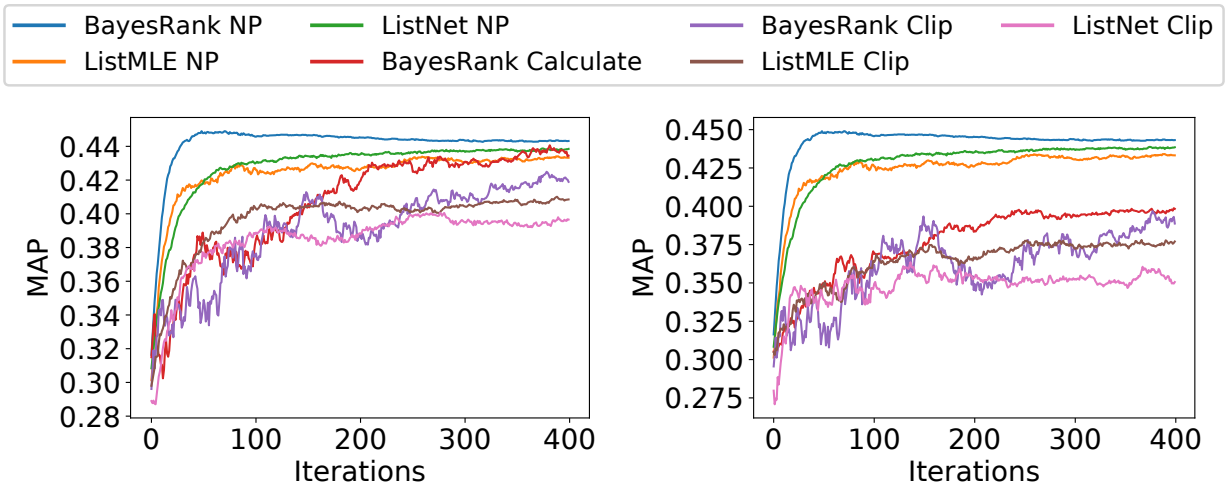
(q) $\sigma = 0.875, \epsilon_{200} = 1.981, \epsilon_{400} = 2.422$

(s) $\sigma = 1.5, \epsilon_{200} = 0.761, \epsilon_{400} = 1.040$

(r) $\sigma = 1.0, \epsilon_{200} = 1.484, \epsilon_{400} = 1.817$

(t) $\sigma = 2.0, \epsilon_{200} = 0.0.551, \epsilon_{400} = 0.758$

Figure 6.1: 5 of 5. Performance on LETOR 4.0 dataset MQ2007, NDCG@8.



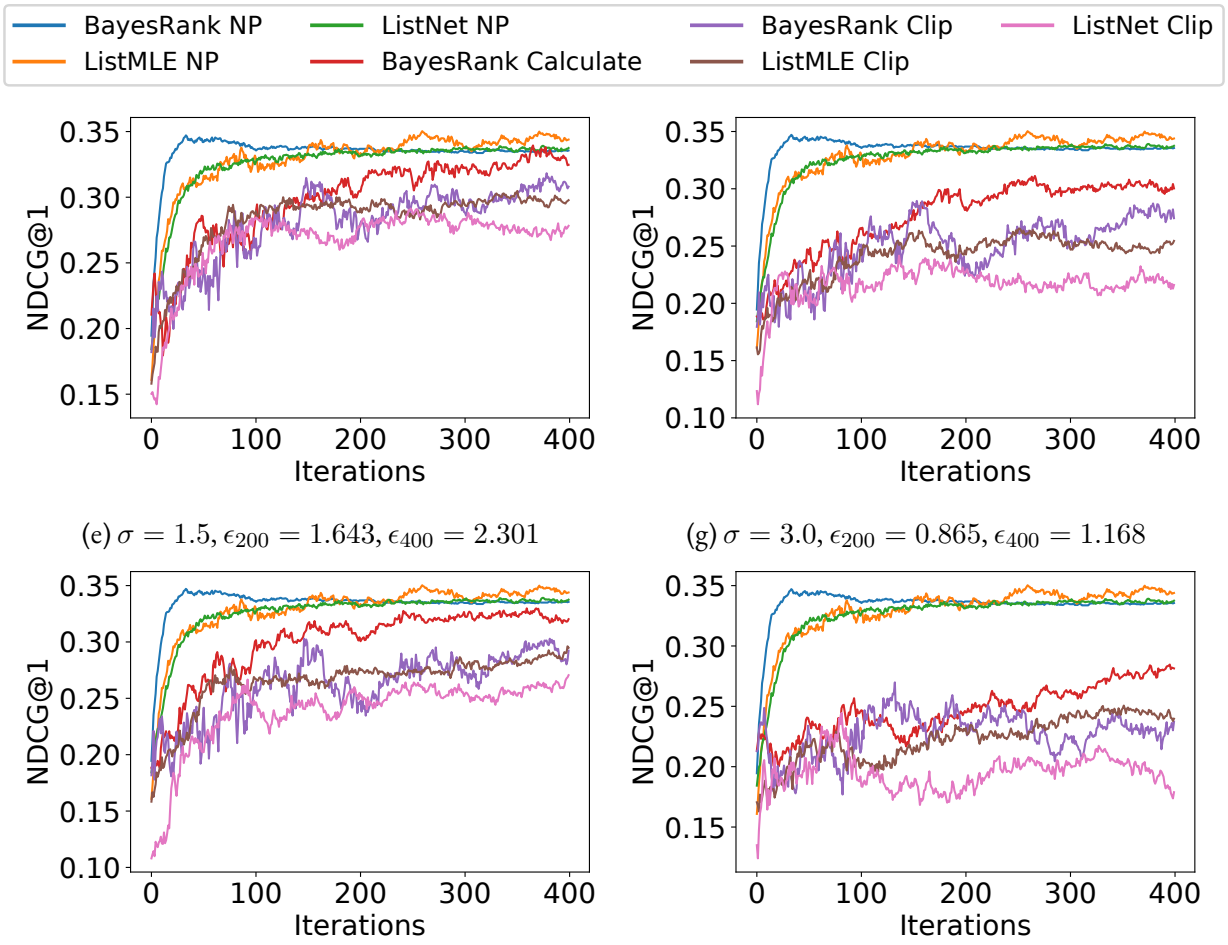
(a) $\sigma = 1.5, \epsilon_{200} = 1.643, \epsilon_{400} = 2.301$

(c) $\sigma = 3.0, \epsilon_{200} = 0.865, \epsilon_{400} = 1.168$

(b) $\sigma = 2.0, \epsilon_{200} = 1.203, \epsilon_{400} = 1.667$

(d) $\sigma = 4.0, \epsilon_{200} = 0.735, \epsilon_{400} = 0.972$

Figure 6.2: 1 of 5. Performance on LETOR 4.0 dataset MQ2008, MAP.



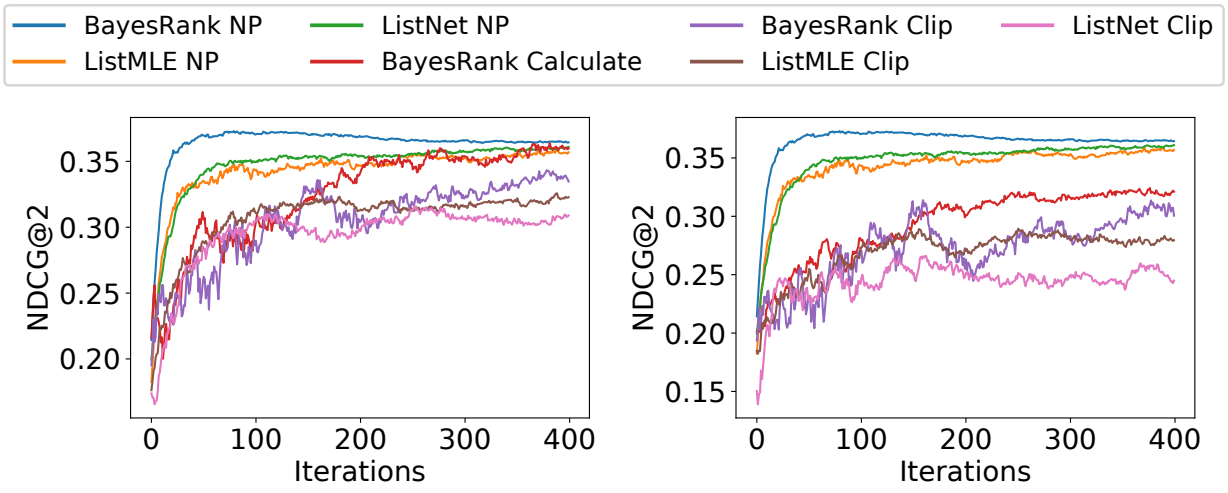
(e) $\sigma = 1.5, \epsilon_{200} = 1.643, \epsilon_{400} = 2.301$

(g) $\sigma = 3.0, \epsilon_{200} = 0.865, \epsilon_{400} = 1.168$

(f) $\sigma = 2.0, \epsilon_{200} = 1.203, \epsilon_{400} = 1.667$

(h) $\sigma = 4.0, \epsilon_{200} = 0.735, \epsilon_{400} = 0.972$

Figure 6.2: 2 of 5. Performance on LETOR 4.0 dataset MQ2008, NDCG@1.



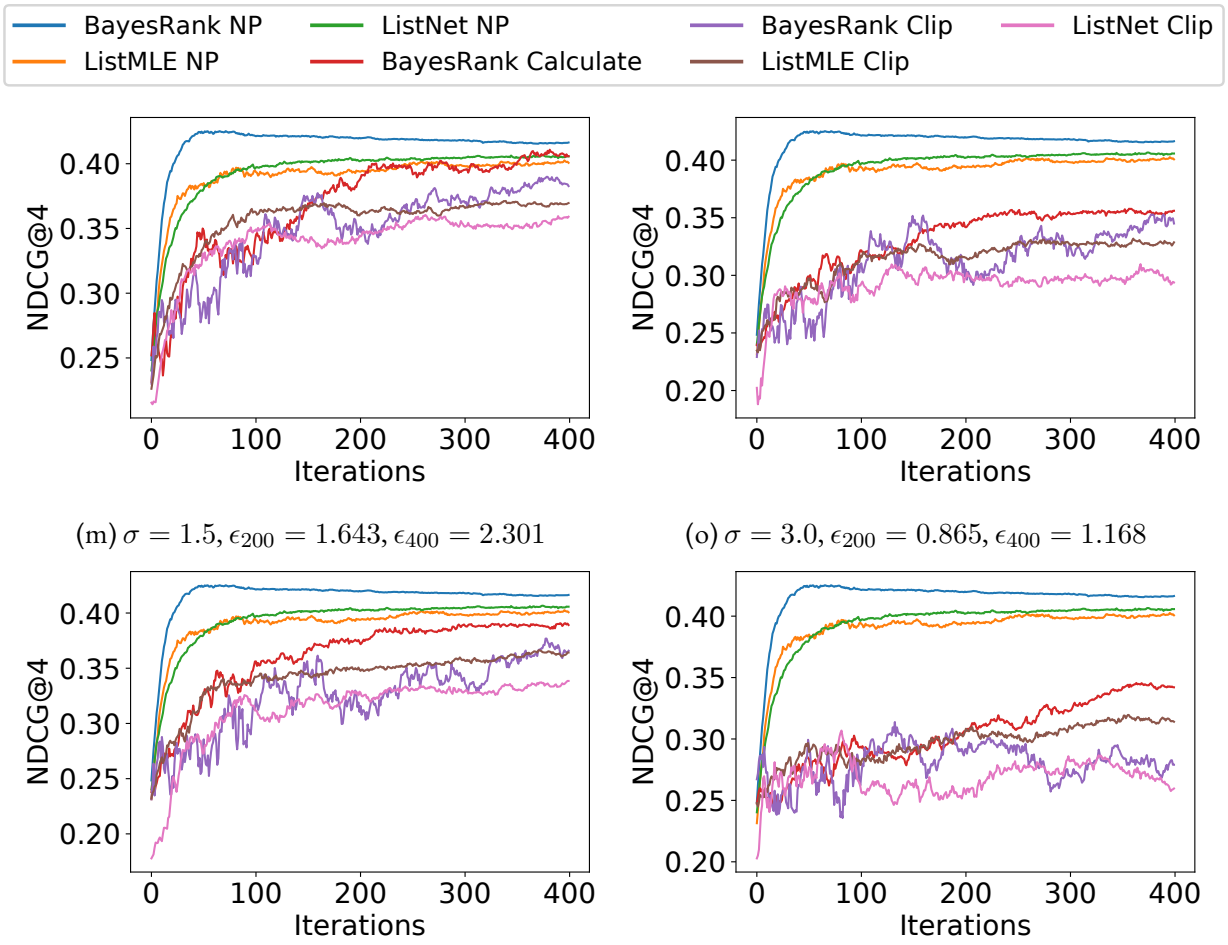
(i) $\sigma = 1.5, \epsilon_{200} = 1.643, \epsilon_{400} = 2.301$

(k) $\sigma = 3.0, \epsilon_{200} = 0.865, \epsilon_{400} = 1.168$

(j) $\sigma = 2.0, \epsilon_{200} = 1.203, \epsilon_{400} = 1.667$

(l) $\sigma = 4.0, \epsilon_{200} = 0.735, \epsilon_{400} = 0.972$

Figure 6.2: 3 of 5. Performance on LETOR 4.0 dataset MQ2008, NDCG@2.



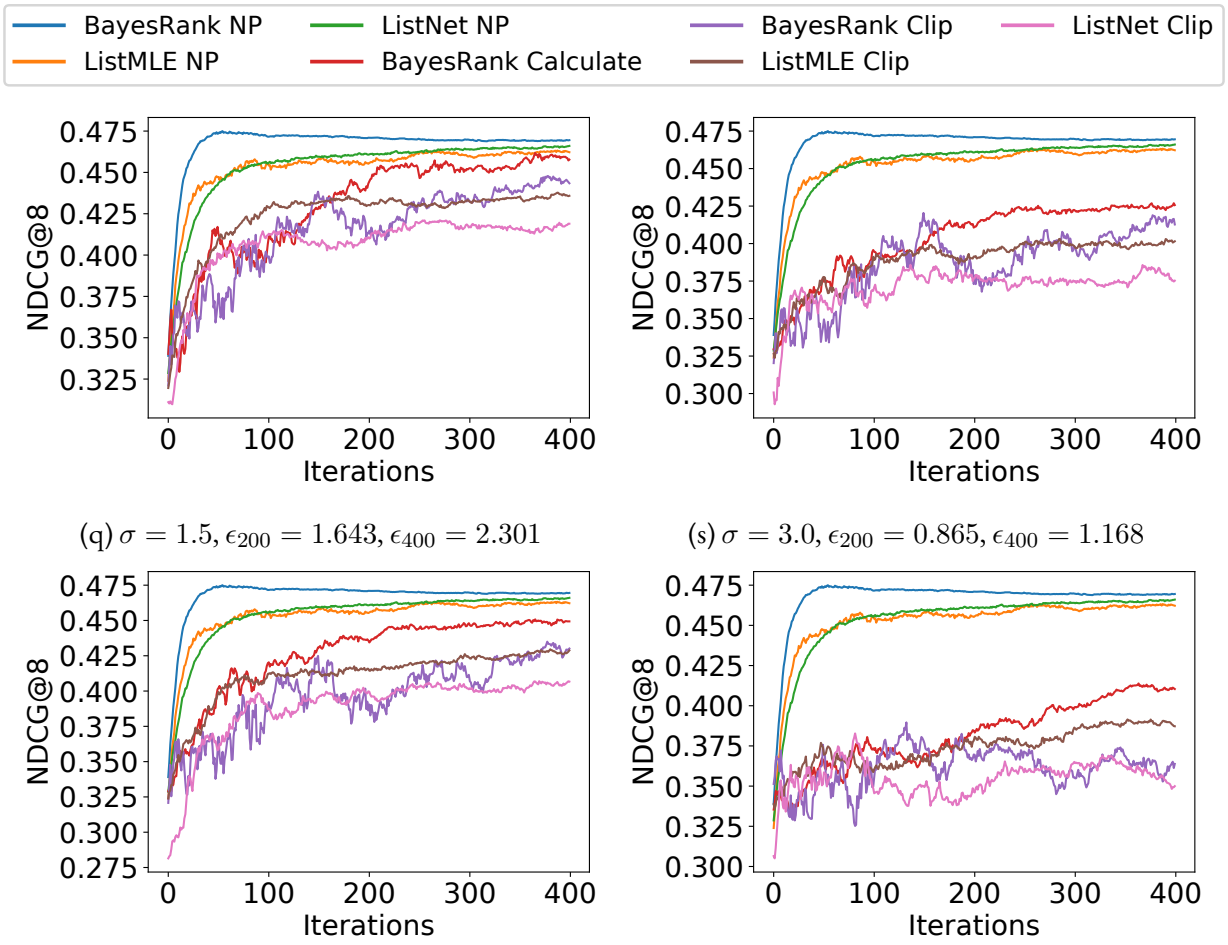
(m) $\sigma = 1.5, \epsilon_{200} = 1.643, \epsilon_{400} = 2.301$

(o) $\sigma = 3.0, \epsilon_{200} = 0.865, \epsilon_{400} = 1.168$

(n) $\sigma = 2.0, \epsilon_{200} = 1.203, \epsilon_{400} = 1.667$

(p) $\sigma = 4.0, \epsilon_{200} = 0.735, \epsilon_{400} = 0.972$

Figure 6.2: 4 of 5. Performance on LETOR 4.0 dataset MQ2008, NDCG@4.



(q) $\sigma = 1.5, \epsilon_{200} = 1.643, \epsilon_{400} = 2.301$

(s) $\sigma = 3.0, \epsilon_{200} = 0.865, \epsilon_{400} = 1.168$

(r) $\sigma = 2.0, \epsilon_{200} = 1.203, \epsilon_{400} = 1.667$

(t) $\sigma = 4.0, \epsilon_{200} = 0.735, \epsilon_{400} = 0.972$

Figure 6.2: 5 of 5. Performance on LETOR 4.0 dataset MQ2008, NDCG@8.

Chapter 7

Conclusion

In this dissertation, several differentially private machine learning algorithms are proposed, in an effort to privately solving an empirical risk minimization problem. Those algorithms apply to different types optimization problems, each with specific conditions, and approaches privacy guarantee through different techniques. The techniques utilize in those algorithms include: sensitivity calculation, periodic averaging, alternating direction method of multipliers, privacy amplification through sub-sampling, sparse vector technique, angle measurement, spectral normalization, etc. Although the algorithms achieve privacy through different ways, they all fall into the recent Rényi differential privacy framework, which means that, they are stronger than the approximate differential privacy definition, and can be converted to the more semantically meaningful (ϵ, δ)-DP with arbitrarily small δ.

The proposed algorithms can be applied on a broad range of machine learning problems. And as the experimental results shows, they demonstrate high utility (in performance), especially in the high privacy region with $\epsilon < 1$. These approaches would greatly help machine learning practitioners, if privacy comes as a serious problem in their application domain.

Bibliography

[1] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[2] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[3] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[4] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[5] Yu-Xiang Wang, Borja Balle, and Shiva Kasiviswanathan. Subsampled r\'enyi differential privacy and analytical moments accountant. *arXiv preprint arXiv:1808.00087*, 2018.

[6] Yuqing Zhu and Yu-Xiang Wang. Poission subsampled rényi differential privacy. In *International Conference on Machine Learning*, pages 7634–7642, 2019.

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

[8] Jiaqi Zhang, Kai Zheng, Wenlong Mou, and Liwei Wang. Efficient private erm for smooth objectives. *arXiv preprint arXiv:1703.09947*, 2017.

[9] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1307–1322. ACM, 2017.

[10] Chen Chen, Jaewoo Lee, and Dan Kifer. Renyi differentially private erm for smooth objectives. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2037–2046, 2019.

[11] Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *arXiv preprint arXiv:0911.5708*, 2009.

[12] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*, pages 25–1, 2012.

[13] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 464–473. IEEE, 2014.

[14] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[15] Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems*, pages 2722–2731, 2017.

[16] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[17] Jaewoo Lee and Daniel Kifer. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1656–1665, 2018.

[18] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[19] Francis Bach. Adaptivity of averaged stochastic gradient descent to local strong convexity for logistic regression. *The Journal of Machine Learning Research*, 15(1):595–627, 2014.

[20] Kunal Talwar, Abhradeep Guha Thakurta, and Li Zhang. Nearly optimal private lasso. In *Advances in Neural Information Processing Systems*, pages 3025–3033, 2015.

[21] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[22] Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Private empirical risk minimization beyond the worst case: The effect of the constraint set geometry. *arXiv preprint arXiv:1411.5417*, 2014.

[23] 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.

[24] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[25] Ernest K Ryu and Stephen Boyd. Primer on monotone operator methods. *Appl. Comput. Math*, 15(1):3–43, 2016.

[26] Dimitri P Bertsekas, W Hager, and O Mangasarian. Nonlinear programming. athena scientific belmont, 2nd edition. *Massachusets, USA*, 2008.

[27] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[28] David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

[29] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[30] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[31] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[32] Moshe Lichman et al. Uci machine learning repository, 2013.

[33] Steven Ruggles, Katie Genadek, Ronald Goeken, Josiah Grover, and Matthew Sobek. Integrated public use microdata series: Version 6.0 [dataset]. *Minneapolis: University of Minnesota*, 23:56, 2015.

[34] S. Hettich and S.D. Bay. The uci dd archive [http://kdd.ics.uci.edu], 1999.

[35] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[36] Olivier Chapelle. Training a support vector machine in the primal. *Neural computation*, 19(5):1155–1178, 2007.

[37] Chen Chen and Jaewoo Lee. Rényi differentially private admm for non-smooth regularized optimization. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, pages 319–328, 2020.

[38] Ryan J Tibshirani, Jonathan Taylor, et al. The solution path of the generalized lasso. *The Annals of Statistics*, 39(3):1335–1371, 2011.

[39] Xiao Zhang, Lingxiao Wang, Yaodong Yu, and Quanquan Gu. A primal-dual analysis of global optimality in nonconvex low-rank matrix recovery. In *International conference on machine learning*, pages 5857–5866, 2018.

[40] Guangcan Liu, Qingshan Liu, and Ping Li. Blessing of dimensionality: Recovering mixture data via dictionary pursuit. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):47–60, 2016.

[41] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[42] Hua Ouyang, Niao He, Long Tran, and Alexander Gray. Stochastic alternating direction method of multipliers. In *International Conference on Machine Learning*, pages 80–88, 2013.

[43] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

[44] Cun-Hui Zhang et al. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942, 2010.

[45] Antti Koskela and Antti Honkela. Learning rate adaptation for differentially private stochastic gradient descent. *arXiv preprint arXiv:1809.03832*, 2018.

[46] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[47] Huachun Tan, Jianshuai Feng, Guangdong Feng, Wuhong Wang, and Yu-Jin Zhang. Traffic volume data outlier recovery via tensor model. *Mathematical Problems in Engineering*, 2013, 2013.

[48] Stanley H Chan, Xiran Wang, and Omar A Elgendy. Plug-and-play admm for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 3(1):84–98, 2016.

[49] Neil K Dhingra, Mihailo R Jovanović, and Zhi-Quan Luo. An admm algorithm for optimal sensor and actuator selection. In *53rd IEEE Conference on Decision and Control*, pages 4039–4044. IEEE, 2014.

[50] Ernie Esser. Applications of lagrangian-based alternating direction methods and connections to split bregman. *CAM report*, 9:31, 2009.

[51] Junfeng Yang and Xiaoming Yuan. Linearized augmented lagrangian and alternating direction methods for nuclear norm minimization. *Mathematics of computation*, 82(281):301–329, 2013.

[52] Puyu Wang and Hai Zhang. Differential privacy for sparse classification learning. *arXiv preprint arXiv:1908.00780*, 2019.

[53] Tao Zhang and Quanyan Zhu. Dynamic differential privacy for admm-based distributed classification learning. *IEEE Transactions on Information Forensics and Security*, 12(1):172–187, 2017.

[54] Xueru Zhang, Mohammad Mahdi Khalili, and Mingyan Liu. Improving the privacy and accuracy of admm-based distributed algorithms. *arXiv preprint arXiv:1806.02246*, 2018.

[55] Zonghao Huang, Rui Hu, Yuanxiong Guo, Eric Chan-Tin, and Yanmin Gong. Dp-admm: Admm-based distributed learning with differential privacy. *IEEE Transactions on Information Forensics and Security*, 2019.

[56] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[57] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

[58] Trevor Park and George Casella. The bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008.

[59] Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y Ng. Efficient l˜ 1 regularized logistic regression. In *AAAI*, volume 6, pages 401–408, 2006.

[60] Mee Young Park and Trevor Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):659–677, 2007.

[61] Yatao An Bian, Xiong Li, Yuncai Liu, and Ming-Hsuan Yang. Parallel coordinate descent newton method for efficient *l*_1-regularized loss minimization. *IEEE transactions on neural networks and learning systems*, 2019.

[62] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[63] Joshua Goodman. Exponential priors for maximum entropy models. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 305–312, 2004.

[64] Yuchen Zhang, Jason D Lee, and Michael I Jordan. l1-regularized neural networks are improperly learnable in polynomial time. In *International Conference on Machine Learning*, pages 993–1001, 2016.

[65] Tao Zhang and Quanyan Zhu. Dynamic differential privacy for admm-based distributed classification learning. *IEEE Transactions on Information Forensics and Security*, 12(1):172–187, 2016.

[66] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10(Dec):2899–2934, 2009.

[67] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[68] Samaneh Azadi, Jiashi Feng, Stefanie Jegelka, and Trevor Darrell. Auxiliary image regularization for deep cnns with noisy labels. *arXiv preprint arXiv:1511.07069*, 2015.

[69] Chen Chen and Jaewoo Lee. Stochastic adaptive line search for differentially private optimization. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020.

[70] 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.

[71] Jian Lou and Yiu-ming Cheung. An uplink communication-efficient approach to featurewise distributed sparse optimization with differential privacy. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[72] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. Differentially private model publishing for deep learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 332–349. IEEE, 2019.

[73] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

[74] Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *Advances in Neural Information Processing Systems*, pages 3727–3740, 2019.

[75] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *arXiv preprint arXiv:1603.01699*, 2016.

[76] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[77] Matteo Sordello and Weijie Su. Robust learning rate selection for stochastic optimization via splitting diagnostic. *arXiv preprint arXiv:1910.08597*, 2019.

[78] Leonard Berrada, Andrew Zisserman, and M Pawan Kumar. Training neural networks for and by interpolation. In *International Conference on Machine Learning*, pages 4758–4768, 2020.

[79] Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. Detecting violations of differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 475–489, 2018.

[80] Raef Bassily, Vitaly Feldman, Kunal Talwar, and Abhradeep Guha Thakurta. Private stochastic convex optimization with optimal rates. In *Advances in Neural Information Processing Systems*, pages 11282–11291, 2019.

[81] Vitaly Feldman, Tomer Koren, and Kunal Talwar. Private stochastic convex optimization: optimal rates in linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 439–449, 2020.

[82] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[83] Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. Privgene: differentially private model fitting using genetic algorithms. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 665–676, 2013.

[84] Tie-Yan Liu. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011.

[85] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, Fabian Nater, and Luc Van Gool. The interestingness of images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1633–1640, 2013.

[86] Oncel Tuzel, Ming-Yu Liu, Yuichi Taguchi, and Arvind Raghunathan. Learning to rank 3d features. In *European Conference on Computer Vision*, pages 520–535. Springer, 2014.

[87] William S Cooper, Fredric C Gey, and Daniel P Dabney. Probabilistic retrieval based on staged logistic regression. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 198–210, 1992.

[88] Ramesh Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71, 2004.

[89] Ping Li, Qiang Wu, and Christopher J Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2008.

[90] Ralf Herbrich. Large margin rank boundaries for ordinal regression. *Advances in large margin classifiers*, pages 115–132, 2000.

[91] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

[92] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, 2006.

[93] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[94] Christopher J Burges, Robert Ragno, and Quoc V Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200, 2007.

[95] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

[96] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in neural information processing systems*, pages 1697–1704, 2008.

[97] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136, 2007.

[98] Tao Qin, Xu-Dong Zhang, Ming-Feng Tsai, De-Sheng Wang, Tie-Yan Liu, and Hang Li. Query-level loss functions for information retrieval. *Information Processing & Management*, 44(2):838–855, 2008.

[99] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, 2008.

[100] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398, 2007.

[101] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 107–114, 2008.

[102] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278, 2007.

[103] Jen-Wei Kuo, Pu-Jen Cheng, and Hsin-Min Wang. Learning to rank from bayesian decision inference. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 827–836, 2009.

[104] Basura Fernando, Efstratios Gavves, Damien Muselet, and Tinne Tuytelaars. Learning to rank based on subsequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2785–2793, 2015.

[105] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[106] Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013.