NOVEL GRADIENT-BASED METHODS FOR DATA DISTRIBUTION AND PRIVACY IN DATA SCIENCE

by

NURDAN KURU

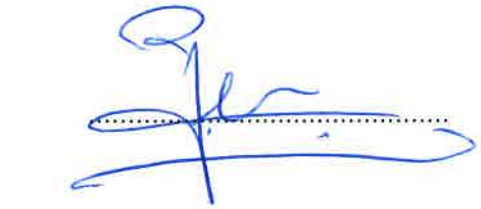
Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Sabancı University
September 2019

NOVEL GRADIENT-BASED METHODS FOR DATA DISTRIBUTION AND PRIVACY IN DATA SCIENCE

APPROVED BY

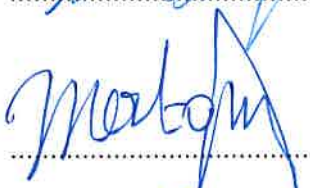
Prof. Dr. Ş. İlker Birbil
(Thesis Supervisor) .....



Prof. Dr. Kerem Bülbül .....

Prof. Dr. A. Taylan Cemgil .....

Assist. Prof. Dr. Mert Gürbüzbalaban .....



Assoc. Prof. Dr. Özgür Martin .....

DATE OF APPROVAL: 20/09/2019

NOVEL GRADIENT-BASED METHODS FOR DATA DISTRIBUTION AND PRIVACY IN DATA SCIENCE

Nurdan Kuru

Industrial Engineering, PhD Thesis, September 2019

Thesis Supervisor: Prof. Dr. Ş. İlker Birbil

Keywords: large-scale optimization, differential privacy, momentum-based algorithms

Abstract

With an increase in the need of storing data at different locations, designing algorithms that can analyze distributed data is becoming more important. In this thesis, we present several gradient-based algorithms, which are customized for data distribution and privacy. First, we propose a provably convergent, second order incremental and inherently parallel algorithm. The proposed algorithm works with distributed data. By using a local quadratic approximation, we achieve to speed-up the convergence with the help of curvature information. We also illustrate that the parallel implementation of our algorithm performs better than a parallel stochastic gradient descent method to solve a large-scale data science problem. This first algorithm solves the problem of using data that resides at different locations. However, this setting is not necessarily enough for data privacy. To guarantee the privacy of the data, we propose differentially private optimization algorithms in the second part of the thesis. The first one among them employs a smoothing approach which is based on using the weighted averages of the history of gradients. This approach helps to decrease the variance of the noise. This reduction in the variance is important for iterative optimization algorithms, since increasing the amount of noise in the algorithm can harm the performance. We also present differentially private version of a recent multistage accelerated algorithm. These extensions use noise related parameter selection and the proposed stepsizes are proportional to the variance of the noisy gradient. The numerical experiments show that our algorithms show a better performance than some well-known differentially private algorithms.

VERİ BİLİMİNDE MAHREMİYET VE VERİ DAĞILIMINA DAYALI GRADYAN TABANLI YENİ METODLAR

Nurdan Kuru

Endüstri Mühendisliği, Doktora Tezi, Eylül 2019

Tez Danışmanı: Prof. Dr. Ş. İlker Birbil

Anahtar Kelimeler: büyük ölçeki eniyileme, diferansiyel mahremiyet, momentum tabanlı algoritmalar

Özet

Veriyi farklı lokasyonlarda saklama ihtiyacındaki artış dağıtık veriyi analiz edebilen algoritmaların önemini de arttırmıştır. Bu tezde, veri dağıtımı ve mahremiyet özelinde tanımlanmış gradyan-tabanlı birkaç algoritma tanıtılacaktır. İlk olarak, ispatlanabilir yakınsak, ikinci dereceden kademeli ve yapısı gereği ayrıştırılabilir bir algoritma önerilecektir. Bu algoritma dağıtık veri ile çalışabilmektedir. Yerel ikinci dereceden yaklaşım kullanarak eğrilik bilgisi yardımıyla yakınsaklığı hızlandırma başarılmıştır. Ek olarak, büyük ölçekli veri bilimi problemi üzerinde tanıtılan algoritmanın paralelleştirilmesinin paralel rassal gradyan inişi algoritmasından daha iyi performans sergilediği gösterilmiştir. Bu tanıtılan ilk algoritma farklı lokasyonlarda saklanan veriyi kullanma problemini çözse de veri güvenliği için yeterli değildir. Tezin ikinci kısmında veri güvenliğini garanti etmek amacıyla diferansiyel olarak mahrem eniyileme algoritmaları tanıtılacaktır. İlk algoritma geçmiş gradyanların ağırlıklı ortalamalarını almaya dayalı bir düzleştirme yaklaşımı kullanmaktadır. Böylelikle gürültü varyansı düşürülmektedir. Varyanstaki bu azalma artan gürültü algoritma performansına zarar verebileceği için eniyileme algoritmaları açısından önemlidir. Ek olarak, yakın zamanda tanıtılmış çok aşamalı hızlandırılmış bir algoritmanın diferansiyel mahrem versiyonu da verilmiştir. Tanıtılan mahrem algoritmaların hepsinin parametre seçimi gürültü göz önünde bulundurularak yapılmış, kullanılan adım boyları gürültü eklenmiş gradyanların varyansı ile orantılı olarak seçilmiştir. Sayısal deneyler de bizim algoritmalarımızın bazı bilinen diferansiyel olarak mahrem algoritmalardan daha iyi performans sergilediğini göstermiştir.

To my family and grandmothers

Acknowledgements

First of all, I would like to express my sincere and deepest gratitude to my advisor Prof. Dr. İlker Birbil, for his guidance, encouragement and constant support. I am thankful to him for all the advice, moral support and patience in guiding me through this thesis. I would like to extend my sincere thanks to my co-advisor Assist. Prof. Dr. Sinan Yıldırım for his guidance, support and important contributions to my study. I am really honored to work with Prof. Dr. İlker Birbil and Assist. Prof. Dr. Sinan Yıldırım.

This dissertation could not be completed without the invaluable input of my other advisers and jury members. I would like to thank Assist. Prof. Dr. Mert Gürbüzbalaban for his support, guidance and hospitality during my visit to Rutgers University. I am also thankful to my thesis committee Prof. Dr. Kerem Bülbül and Prof. Dr. Ali Taylan Cemgil for their valuable time, interest and insightful comments. Special thanks to Assoc. Prof. Dr. Murat Çokol for being an advisor and collaborator to me. I have learnt a lot from him about computational biology.

I am grateful to all of my friends from Industrial Engineering and Mathematics Programs. Their invaluable friendship made me feel at home at Sabancı University. Specially, I want to thank to my dear friends Esra Gül, Tekgül Kalaycı and Gamze Kuruk for always being there for me.

I am deeply grateful to my mother Lütfiye, my father İbrahim and my sister Nurcan. Without their support and patience, I could never complete this thesis. They were always next to me with all their warmness and unwavering love. I also want to thank to my grandparents for their endless love. Although they are not with us anymore, I still feel my grandmoms' support behind me.

Lastly, I would like to thank TÜBİTAK for supporting me financially by granting a scholarship for my visit to Rutgers University.

Table of Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

With the recent interest in maintaining data at different locations, the need for the analysis of distributed data has increased. Many organizations prefer to distribute their data since it is more secure and less costly. Moreover, accessing distributed data is speedy and even if there is a failure about a part of the data, the other parts are not affected. This scenario considers the distribution of data at one organization. It is also possible that multiple companies desire a common analysis for their similar data. In this case, it is not practical to collect all data in a single location, so the first aim is to find a way to use the distributed data without changing its location. Many studies in the literature deal with this problem. In this thesis, we present a new algorithm, which not only uses distributed data but is also provably convergent, second order incremental and inherently parallel. This algorithm does not require moving data from one location to another, but if the data holders have a privacy concern, then the proposed algorithm does not necessarily guarantee privacy.

Nowadays, when entering a website or watching a movie, people provide information about themselves. Even if this data is shared voluntarily, the main issue for a company while using this customer information is to make sure that the identity of any individual is not revealed. To solve this security problem, we use differential privacy, one of the popular approaches in machine learning. Differential privacy means constructing a mechanism that does not give any clue about whether information related to an individual is present or not present in data. In other words, the output of the mechanism does not change probabilistically when a new individual's information is added. Privacy does not require restricting the questions that can be asked about

data or anonymization of data. Rather, it requires that the adversary has no more information about an individual after substracting this individual's information.

Differential privacy is also popular in the optimization context. It is known that by using a suitable noise adding mechanism, iterative algorithms can be made differentially private [4, 76]. However, the privacy level can easily be harmed by this process since a new question is answered by using the same data continuously over subsequent iterations. We propose adjustments to improve the privacy level to make differential privacy more compatible with iterative optimization algorithms. Some of these adjustments, such as momentum and averaging, have already been used to speed up the gradient descent (GD) and the stochastic gradient descent (SGD) algorithms [61, 80].

Our aim is to obtain improved differentially private algorithms by employing gradient averaging which is used in momentum-based algorithms such as Polyak's heavy ball (HB) [66] and Nesterov's accelerated gradient descent (NAG) [59]. With this aim, we propose two differentially private algorithms based on HB and NAG, and their stochastic versions.

Our first algorithm, called the differentially private gradient descent algorithm with smoothing (DP-GDwS), employs a smoothing approach and uses the information from the previous iterations while taking steps. We have constructed this smoothing mechanism with the aim of improving the privacy level by taking the weighted averages of the current and the previous noisy gradients. While studying the algorithm's convergence, we have observed that it can be analyzed in the form of HB. Thus, to further give its convergence rate, we have made use of the dynamical system approach, which is one of the recent approaches for analyzing the first order methods [30, 40, 48]. Our second deterministic algorithm, the differentially private multistage accelerated gradient (DP-MAG), is based on the multistage accelerated stochastic gradient algorithm (M-ASG) introduced in [3]. The authors have proposed and analyzed an accelerated method dealing with noisy gradients, but do not consider the differential privacy in the algorithm design. Moreover, the variance of the added noise is determined without taking the data into account. We introduce the private version of this algorithm with parameters compatible with the differential privacy noise. Additionally, we use a noise dividing scheme by considering the stages of the algorithm. By following the same steps as in [3], we present the convergence analysis of our new multi-stage algorithm.

In the final part, we propose the stochastic versions of DP-GDwS and DP-MAG. These versions are based on subsampling at each iteration instead of using complete data. It is known from the literature that subsampling has an amplification effect over differential privacy because of the randomness of data selection. With the aim of improving the algorithm and decreasing the amount of noise, we construct the stochastic versions of our algorithms with different sample sizes and show their advantages with a numerical study.

1.1 Motivation

In many areas, the use of distributed data, and the need for analyzing data collected from different owners are increasing. To meet this expectation, we present an improvement over an earlier version of our algorithm HAMSI (Hessian Approximated Multiple Subsets Iteration), which is a generic second order incremental algorithm for solving large-scale partially separable convex and nonconvex optimization problems over distributed data. Our motivation is to improve the convergence proof and the performance of the algorithm. The new version of the analysis is stronger and easier to follow with a simplified notation. To the best of our knowledge, a proof for such a deterministic algorithm has not been given before in the machine learning literature. We also present the stochastic version of HAMSI and provide its convergence proof. Moreover, after investigating a few shared-memory parallelization techniques, we present a load balancing heuristic that results in a better numerical performance. Thus, we obtain an algorithm that is provably convergent, has better performance than parallel SGD and works well on distributed data. The presented algorithm solves the problem of using data distributed to multiple locations, but the data owners may also want to protect the privacy of their data. However, HAMSI does not necessarily guarantee data privacy in the sense of differential privacy.

Privacy becomes an important concern with the increasing need of collection and analysis of data. Accordingly, the privacy concern of optimization algorithms has become a popular problem recently. Despite this attention in the literature, a need remains for further research and new ideas. The existing studies dealing with the privacy of iterative algorithms perturb the input or output by using a suitable noise

adding mechanism. However, since each new iteration harms privacy, finding a way to decrease the noise or improve the privacy level by adding the same amount of noise are important contributions for this concept. The first differentially private SGD was introduced in 2013. Although various problems have been solved privately using similar ideas, there still exists a need for further research, especially on the parameter selection of algorithms by considering the effect of random noise. Moreover, the performance and analysis of accelerated methods under privacy context has not been throughly discussed in the literature.

1.2 Problem Description

A vast variety of problems in machine learning can be written as unconstrained optimization problems of the form

$$\min_{x} \sum_{i \in \mathcal{I}} f_i(x), \tag{1.1}$$

where x is a parameter vector and f_i are a collection of functions. Each i in $\mathcal{I}$ represents a single function and the number of additive terms in the objective function gives the size of data. This is the general form of the function that we consider during this study. For different concepts, there will be various assumptions over the function f and the other parameters.

In this thesis, our first aim is to present an efficient algorithm to solve a large scale problem on distributed data. A natural approach for solving this type of problems is applying parallel computation and a divide-and-conquer approach. When the problem is *separable*, divide-and-conquer is an advantageous way to obtain a solution to the problem. However, separability is not satisfied for many type of problems such as various matrix decomposition or regression problems. Fortunately, this type of problems have some inherent partially separable structure which allows to run the algorithm in parallel. Specifically, we focus on the following optimization problem:

$$\min_{x \in \mathbb{R}^{|\mathcal{J}|}} \sum_{i \in \mathcal{I}} f_i(x_{\boldsymbol{\alpha}_i}), \tag{1.2}$$

where each term $f_i : \mathbb{R}^{|\mathcal{J}|} \mapsto \mathbb{R}$ for $i \in \mathcal{I}$ of the overall objective function f is twice continuously differentiable. Because of the large-scale problem, $\mathcal{I}$ has a large cardinality where $\mathcal{I} \equiv \{1, 2, \ldots, |\mathcal{I}|\}$. On the other hand, each term f_i can be written as $f_i(x) \equiv$

$f_i(x_{\boldsymbol{\alpha}_i})$ since it depends only on a small subset of the elements of x. We use $\boldsymbol{\alpha}_i$ as index sets such that $\boldsymbol{\alpha}_i \subseteq \mathcal{J}$ for all $i \in \mathcal{I}$ and the function index set $\mathcal{J} \equiv \{1, 2, \ldots, |\mathcal{J}|\}$. Each singleton $j \in \mathcal{J}$ is denoted as x_j and corresponds to a unique component of vector x. In other words, when we have $\boldsymbol{\alpha} = \{j_1, j_2, \ldots, j_A\}$, the related vector is $x_{\boldsymbol{\alpha}} = (x_{j_1}, x_{j_1}, \ldots, x_{j_A})$.

To solve our partially separable problem, we use the idea of *incremental methods* (*cf.* [10]). At each iteration τ, a subset of the function terms f_i is chosen by the incremental method. That is, a subset $S^{(\tau)}$ is selected from the function indices such that $S^{(\tau)} \subset \mathcal{I}$. In the first version of our algorithm, we follow a deterministic framework for the selection of the subsets. For stochastic HAMSI, this selection is done randomly. The convergence analyses for both algorithms are provided in Chapter 3. Here, both in deterministic and stochastic version, by careful selection of subsets $S^{(\tau)}$ at each step τ, the proxy objective satisfies separability and hence, parallel computation can be applied.

For the second part of this thesis, our main focus is to present private optimization algorithms with an improved performance. With this aim, we again deal with the problem in (1.1) with an additional assumption of strongly convexity. Thus, we can update the objective function as

$$\min_{x} \sum_{i \in \mathcal{I}} f_i(x) + \lambda \|x\|^2, \tag{1.3}$$

where λ is the regularization constant, f is a twice continuously differentiable function. In this part, since the main focus is protecting the privacy, we employ differential privacy approach. Differential privacy can be achieved by adding noise to the data, to the function, or to the iteration vector. In a setting like ours, the iteration vector is revealed at intermediate steps. Therefore, a suitable noise vector should be added to the gradient or iteration vector at each step. However, this noise does harm the performance of the algorithm in such a way that it may even cause divergence. Here, we aim to preserve the privacy of the data while maintaining the performance of the algorithm. We mainly work on accelerated first order algorithms which are based on using of not only the current gradient but also gradients from previous iterations to take the next step. We aim to determine the parameters of the presented algorithms compatible with the added random noise. Especially the classical stepsize formulations

used in non-private settings may affect the performance of the algorithms negatively after adding the noise required for differential privacy. We propose two differentially private algorithms based on HB and NAG with special stepsize formulation in both deterministic and stochastic settings. In this part of the thesis, the deterministic setting corresponds to using of complete data while computing the gradient and the stochastic setting is based on sampling of data. With the use of stochastic gradients, we improve the performance as a result of the decrease in the noise variance.

1.3 Proposed Approaches

The main problem in the first part of this study is using of distributed data in an optimization algorithm. To this end, we consider our algorithm HAMSI. This algorithm is inherently parallel, based on a local quadratic approximation, and thus, contain curvature information which helps to speed-up the convergence. In its original form, an analysis is given related to the convergence of HAMSI by proving only that the limit of the infimum of the gradient converges to zero. In Section 3.1, we provide an improved convergence analysis showing the limit of the gradient tends to zero as the number of iterations increases. The new analysis is both stronger and easier to follow with a simplified notation. For further improvement, we consider the stochastic setting which is a popular approach for the design of gradient-based algorithms. We present stochastic HAMSI which is constructed by random selection of subsets at each iteration. The convergence analysis for this version is also given in Chapter 3. In the numerical experiments, we combine HAMSI with some parallelization techniques and use the L-BFGS implementation to obtain approximate Hessian matrices. In this framework, when a large-scale matrix factorization problem is solved for both HAMSI and a parallel gradient descent method, the numerical experiments show that HAMSI converges more rapidly than the other algorithm.

Secondly, we propose differentially private version of some first order optimization algorithms and analyze their convergence properties. We first present a smoothing approach to decrease the amount of noise required for the privacy of GD. Like other differentially private methods in the literature, this approach also perturbs the output at every iteration by adding noise to the approximate gradient. However, unlike those

methods, the proposed approach uses exponential smoothing to obtain a weighted sum of the past and most recent approximate gradients. This weighting mechanism allows us to run the resulting gradient descent algorithm for a large number of iterations without breaching privacy. On the other hand, the selection of parameters such as stepsize is also an important part of the algorithm design. As the last stage, we also analyze the convergence behavior of our algorithm by using the dynamical system approach.

The second deterministic differentially private algorithm we propose is an updated version of the algorihtm M-ASG. M-ASG is a multistage accelerated algorithm and it uses noisy gradients at each iteration. Although M-ASG uses noisy gradients, its differential privacy is not considered before. We improve this algorithm in the privacy context by using a noise related stepsize formula and a special noise dividing mechanism. The convergence analysis for the original algorithm has already been given in [3]. We also analyze our version by following the same steps.

In the last part of this thesis, we present the stochastic versions of the two differentially private algorithms with the aim of further improvement of their performances. The amplification effect of subsampling over the noise variance has already been studied in the literature. Thus, instead of using the entire data at each iteration, we take random samples from the data. The numerical experiments also support our findings and we achieve an improvement over the performance of some known differentially private optimization algorithms.

1.4 Contributions

The contributions of this thesis to the scientific literature can be summarized in two parts. We can list the contributions in the first part as follows:

- We propose an improved version of our algorithm HAMSI [45] which is distributed, second order incremental and inherently parallel. We also provide a stronger result for the convergence analysis of this algorithm.
- We propose a stochastic version of HAMSI and analyze its convergence.
- By applying several parallelization techniques and presenting a simple load balancing heuristic, we obtain a better numerical performance than the earlier ver-

sion.

The distributed data concept brings another problem of privacy. In case data comes from different users, the data holders be engaged in a common analysis while securing their data. Starting with this idea, we reserve the second part of the thesis for privacy concern and present private first order optimization algorithms by using differential privacy. Our further contributions with the second part are listed below:

- The first deterministic method we propose is a differentially private algorithm based on using weighted averages of current and previous gradients. We call this approach as *smoothing*. The presented algorithm is a special case of Polyak's heavy ball method. By using this fact, we establish the relationship between the convergence rate and the algorithm parameters by using the dynamical system approach.

- The second deterministic method is a differentially private version of the recent algorithm Multistage Accelerated Stochastic Gradient (M-ASG) algorithm [3]. This algorithm is based on using accelerated gradient method at each iteration, and different from the existing literature, they use a stage-related stepsize formula. The gradient contains noise in [3] as well. However, their noise is not constructed with the aim of privacy. Here, we present a differentially private version of M-ASG by selecting the algorithm parameters compatible with differential privacy noise.

- The algorithms that we classify as stochastic in the second part are constructed by using sampling of data. It is known from the literature that the randomness coming from sampling of data helps to decrease the noise variance. With this idea, we present the stochastic versions of the differentially private gradient descent with smoothing (DP-GDwS) and the differentially private multistage accelerated gradient (DP-MAG) algorithms.

- We use a special stepsize formula for all of the presented differentially private algorithms. By taking into account their original stepsize formulation presented for non-private setting, we add a term related to the random noise.

1.5 Overview of The Thesis

This thesis consists of four chapters including the introduction. The second chapter is reserved for a literature review to explain the place of this thesis in the literature.
We consider two different concepts and problems over the solution of (1.1), so divide the main study into two parts which are presented in Chapter 3 and Chapter 4. In the first part, we propose an improved version of our algorithm. The second part starting from Chapter 4 is based on differential privacy concept. We consider the privacy issue of the first order optimization algorithms and aim to find a way to improve the performance of these algorithms under privacy noise. Thus, we propose two algorithms based on heavy ball and accelerated gradient methods. The first part of this chapter is reserved for deterministic versions of these algorithms with convergence rate analysis. In the last part, we introduce their stochastic versions and provide numerical experiments.

Chapter 2

Literature Review

We group the related literature under three sections: data distribution, differential privacy and first order accelerated optimization algorithms. First part lists the studies that give us an insight to propose HAMSI. The differential privacy studies, especially from optimization point of view are given in the second section. The last section is reserved for momentum-based algorithms and their convergence analysis for convex problems.

2.1 Data Distribution

In the first chapter of this thesis, we present an incremental and parallel second-order algorithm which uses approximate curvature information to solve distributed large-scale problems. We experience that using second order information can accelerate convergence even with the incremental gradient as in our case. To obtain the curvature information, we model the local approximations by quadratic functions. With the help of the structure of our objective function, we characterize it as a bipartite graph and the gradient is evaluated for a choosen subset of the component functions at each iteration. This is similar to incremental and aggregate methods [10, 11, 68, 75, 80]. The subsets of component functions are choosen by considering the separability of inner problems. This separability structure allows us to distribute the computations over multiple processors and do the stepwise computations in parallel. Thus, our algorithm makes use of modern distributed and multicore computer systems easily.

A similar distribution scheme is introduced before in [34] for matrix factorization

problem. The authors use this setting to deal with large-scale distributed data while taking advantage of its partially separability nature [19, 73]. The problem is solved by using SGD and the convergence analysis is provided. The second order incremental methods are also studied before in the literature. For the least squares problem, a similar algorithm which is an incremental version of Gauss-Newton method is presented in [9]. The generalization of this approach is studied as well [38]. They prove the linear convergence of the method under the assumptions of strong convexity and *gradient growth*. Furthermore, the inversion of the exact Hessian matrices of the component functions is required for their method. Another distributed Newton-like method is presented for convex problems in [71]. Their approach requires the computation and inversion of Hessian matrices local to each node. The setting of this method does not allow space decomposition, thus the *entire* parameter vector should be stored in memory and communicated at every iteration. In [74], Sohl-Dickstein et. al introduce an incremental aggregated quasi-Newton algorithm based on updating the quadratic model of one component function at each iteration.

The stochastic versions of Quasi-Newton methods form another group of work in the literature. Gower et. al propose the stochastic block BFGS which does multi-secant updates and achieves linear convergence for the solution of convex problems by using variance-reduced gradients [37]. Stochastic quasi-Newton methods with linear convergence rates are presented in [56] under convexity assumptions. Their algorithm uses aggregated gradients and variance reduction techniques. Although it is known that these methods are useful for certain applications, their aggregation steps make them incompatible with parallel computation. Yousefian et. al propose a regularized stochastic quasi-Newton method under *merely* convexity assumption [89]. Note that because of the difficulties of applying a quasi-Newton method with stochastic (or incremental) gradients, there exists a *data consistency* problem in this setting, which causes a suitable structure for parallel computation [8, 14, 70]. The stochastic variants of Quasi-Newton methods is still a popular subject with recent studies [7, 36, 42, 87].

In [38], Gürbüzbalaban et. al analyze the second order incremental methods, but because of their convexity assumption, their analysis does not cover our *deterministic algorithm*. Deterministic HAMSI is analyzed by following [75]. The analysis given in [75] is related to the incremental gradient algorithms for nonconvex problems and

not exactly the same with our approach since they do not consider the incorporation of second order information. An analysis for stochastic quasi-Newton methods with nonconvex objective function is given in [85]. By considering the objective function as an expected value expression with a discrete probability distribution, their analysis is valid for our stochastic algorithm. However, the direct analysis of our stochastic algorithm is based on the paper [12] which studies online learning algorithms under different assumptions. In a more recent study, Mokhtari et. al consider an incremental stochastic quasi-Newton method and give its convergence properties [55]. They prove the superlinear convergence rate of their algorithm which is a stochastic version of the L-BFGS method.

2.2 Differential Privacy

Differential privacy is first introduced by Dwork as a solution to the problem of revealing useful information about a data without harming privacy of any individual [23]. In this study, a mechanism satisfying ϵ-differentially privacy is also introduced. This mechanism adds a suitable noise to the answer of the query applied to data. Although it is presented as ϵ-indistinguishability, the first definition and mechanism design of differential privacy is based on [26]. In [26], it is also shown that Laplace mechanism satisfies differential privacy. Later, another mechanism complement to Laplace, *exponential mechanism* is presented by McSherry and Talwar [53]. Similar to studies [23,26], we use Laplace noise in our design. Although ϵ-DP is preferred to protect data privacy, a relaxed version, (ϵ, δ)-differential privacy is also preferred in some studies [62]. This version is suitable when ϵ-DP is too strict and prevent to obtain a meaningful result. For our algorithms, we prefer ϵ-DP. A lot of variants of differential privacy are also studied in the literature such as Renyi differential privacy [54], concentrated differential privacy [28], local differential privacy [22]. A recent survey studies these variants which provides different types of privacy guarantees [21].

After constructing a differentially private mechanism, one of the basic problems is the effect of composition over the privacy level. Since we aim to obtain differentially private iterative optimization algorithm, the effect of composition is a crucial component of our algortithm design. As an answer to this problem, [25, 26] gives a bound as

$k\epsilon$-DP for k times composition of an ϵ-DP mechanism. Later, it is proven that tighter bounds for composition are possible with advanced composition theorems [13,28,29,43].

Differential privacy is a popular approach for the design of privacy-preserving machine learning algorithms. The idea of differential privacy is used for protecting data privacy for many type of problems such as boosting [29], linear and logistic regression [17,92], support vector machines (SVM) [69] and risk minimization [18]. Especially, there is a large literature about the differential privacy of emprical risk minimization (ERM) [4, 18, 47, 93]. This is not suprising since differential privacy is popular in machine learning models and ERM covers many machine learning tasks. The first known study about differential privacy of ERM is presented by Chaudhuri et. al [18]. They present two algorithms the first of which is based on output perturbation and the noise is added to the output of ERM algorithm. With the second algorithm, they introduce objective perturbation and add the noise onto the objective function before minimizing. In this paper, some open problems are mentioned related to extending the objective perturbation idea to the general convex functions. One of the problems is answered by Kifer et. al [47] in 2012. They modify the idea to general convex objectives such that the required noise is smaller than [18]. In another study [4], Bassily et. al consider the differential privacy of convex ERM problem and present an efficient exponential mechanism satisfying ϵ-DP. Moreover, they obtain improved bounds than [47] and [18], and show that their algorithms match the lower bounds. In the same study, an algorithm which hits the lower bounds for strongly convex loss function is also provided. Another study, [79] gives better utility bounds for problems such as sparse linear regression. In 2017, Zhang et. al [93] present two efficient algorithms with privacy and utility guarantees. The algorithms at the previously mentioned papers [4, 79] are slow and require to run the model for $\Omega(n^2)$ iterations to satisfy a predefined accuracy where n is the data size. In [93], they eliminate this condition and show that their algorithm on strongly convex and smooth objective is much faster than differentially private SGD algorithm presented in [4]. In the same paper, a random round SGD algorithm is introduced for non-convex and smooth objectives. In 2017, Wang et. al [83] propose algorithms which achive optimal or near optimal utility bounds. Their algorithm design is based on gradient perturbation and Gaussian mechanism. In the second part, they consider a non-convex objective function and obtain a tighter utility bound than [93]. There

also exists some recent papers about the differential privacy of ERM in the literature [44,82,84]. In [84], the authors propose a differentially private SGD for nonconvex ERM and analyze privacy and utility guarantees. The distributed version of the proposed algorithm is also given. In the second paper, Wang et. al propose DP Laplacian smoothing SGD algorithm in convex and nonconvex settings. The algorithm is based on Laplacian operator which is introduced in [63] to improve the performance by reducing the variance of SGD. The proposed algorithm is compared with DP-SGD and obtained a better performance for logistic regression and SVM. Although most of the mentioned studies in ERM deal with convex and strongly convex problems, differentially private algorithms for non-convex objective function (especially in deep learning) are also studied in the literature [1, 72, 90]. Similar to Abadi et. al [1], we will use the idea of norm clipping to bound the gradient. The idea of norm clipping is based on scaling down the norm of the gradient to a threshold value C if it is greater than C.

Specifically, we are dealing with differentially private gradient-based algorithm design. To explain the differences of our work from others, we further discuss the existing studies related to differential privacy of GD and SGD in detail. In [76], the authors solve logistic regression problem in a differentially private setting constructed by using SGD and mini-batch SGD. At each iteration, they perturb the gradient by adding a suitable noise. Our methods can be thought as generalizations of this approach, since there is a local privacy at each iteration in our approach as well. However, their method has a disadvantage since the number of iterations is more restricted than ours because of higher variance. Another study that introduces the differentially private version of SGD is [17]. In this paper, they solve logistic regression problem and reveal the identifier by adding a suitable random noise. This approach performs better than the one presented by Dwork [26]. However, it does not guarantee the privacy of each iteration, so not compatible with our scenario. Song et. al add heterogeneous noise while learning with SGD and claim that the performance is better than the less noisy and single learning rate algorithms [77]. In [4], the authors add a suitable random noise to the gradient calculation of stochastic variant of the gradient descent algorithm. By taking help of subsampling, the privacy level is improved as in our approach. Although stochastic algorithms are more popular in the differential privacy context, there still exist studies considering the differentially private deterministic algorithms. Again in [4],

the deterministic version of their algorithm which uses gradient descent is also considered. As a related study, Zhang et. al present two algorithms in their study [93]. In addition to the other differences with our algorithm, they focus on (ϵ, δ)-differential privacy which is a weaker form of ϵ-differential privacy. In a more recent study [65], Pichapati et. al consider a differentially private SGD algorithm and to decrease the amount of noise required for privacy they use coordinate-wise adaptive clipping. Their idea is based on clipping the gradient by using its mean and variance. Again, the weaker form, (ϵ, δ)-differential privacy, is satisfied for their algorithm.

In this thesis, we take advantage of subsampling while preserving the privacy of data. It is known that using sample batches instead of a single point is a more reliable way for SGD although it is more costly than the single point selection. Not only for optimization algorithms, also in differential privacy context, the improving effect of sampling has been studied. [49] prove the amplification effect of sampling and give a new privacy bound resulting from this sampling procedure. This proof handles with the privacy of the data whose one entity is taken out. but in our case, we are changing one element of data with a new entity, so this study and the related theorem is not completely applicable to our case. In [86], Wang et. al also deal with subsampling and differential privacy, and provide a theorem and proof based on the ideas presented in [6]. We use a different version of this theorem that we have obtained by making the privacy level tighter based on the same proof.

Another arrangement used in our algorithm, DP-GDwS is taking the step at each iteration by using not only the current gradient also the information coming from the previous iteration. Although it is novel to use this approach in the context of differential privacy, there are studies which use a similar idea to speed-up the convergence of GD and SGD. The detailed review of accelerated methods is given in the next section.

2.3 First Order Accelerated Algorithms

First order algorithms have been used since 1950s in the context of convex optimization [31]. They are still popular for many types of problems arising in machine learning, control theory and signal processing [5, 50, 57] because of small cost per-iteration and being compatible with large scale optimization problems [35]. Especially, when com-

puting the second order information becomes computationally expensive, first order algorithms are prefarable. The simplest first order algorithm, Gradient Descent [16] achieves linear convergence with suitable stepsize when the objective is strongly convex. In [58], it is proved that the convergence rate of first order algorithms on convex problems with Lipschitz continous gradient cannot achieve a better rate than $O(1/k^2)$ where k is the number of iterations. Nesterov [59] presents an accelerated algorithm (NAG) that converges with rate $O(1/k^2)$ which closes the gap between the guaranteed optimal rate and the convergence rate that is achieved for GD. The NAG method is based on using a momentum term to accelerate the algorithm. That is, accelerated methods use not only the current but also the previous iterations and the corresponding gradients to take a step. Another popular momentum-based algorithm is Polyak's Heavy Ball (HB) method. Different from the NAG method, HB method uses the previous iteration but not the previous gradient to take a step. Nesterov's method is known as achieving linear rate for strongly convex objective. On the other hand the HB method converges linearly and has a better convergence factor for strongly convex, twice continuously differentiable objective and Lipschitz continous gradient, in other words, it is better than the GD and NAG methods under these assumptions. For not necessarily convex and Lipschitz continous gradient, the necessary conditions for the convergence of the HB method is given in the literature [91]. On the other hand, [48] shows with an example that the HB method may not converge under strong convexity.

There are many studies in the literature dealing with the convergence of the momentum-based first order optimization methods. In [35], Ghadimi et al. present a global convergence analysis for HB method on convex optimization problems. When objective function is convex and Lipschitz continous, they prove $O(1/k)$ convergence for the HB method where k is the number of iterations. Under strong convexity assumption, it is proved that the algorithm converges linearly to the minimum. Our DP-GDwS algorithm is based on the HB method with special parameters and we solve a strongly convex problem as well. On the other hand, because of the noisy gradients in our method, this analysis cannot be followed.

Although most of the papers in the literature take deterministic setting into account while analyzing the momentum-based methods, there exist some studies which analyze the convergence of stochastic accelerated algorithms [32, 51, 67, 88]. In [32],

Gadat et. al present almost sure convergence result for stochastic heavy ball method for nonconvex coercive functions and give convergence rate analysis for strongly convex quadratic functions. Yang et. al present a unified framework for the analysis of the GD, the NAG and the HB methods in stochastic setting for both convex and strongly convex objectives [88]. They use constant stepsize and assume the boundedness of the variance of the noise which is defined as the difference between full gradient and stochastic gradient. In [88], again a unified framework similar to [88] is presented with an additional nonconvexity assumption. Similarly, the variance is assumed bounded in this paper. Both papers achieve a rate of $O(1/\sqrt{k})$ by analyzing the Cesaro averages of the iterations. Another study [67] analyzes the stability and generalization error of stochastic gradient with the momentum method. In [51], the authors deal with the stochastic heavy ball method and presents the first linear convergence result in a simplified setting of quadratic objectives. Jain et al. introduce a stochastic variant of the NAG method for the solution of least squares regression [41]. It is known that the excess risk of the algorithm can be separated as bias and variance terms. In this paper, they achieve a better rate for the bias while retaining minimax rate for the variance term. On the other hand, in [46], they claim that although accelerated methods beats the SGD method in deterministic setting, that is not the case for stochastic approximation. They prove this claim by solving simple problems with the best parameter setting and compare the SGD method with the NAG and the HB methods. They also present an algorithm called Accelerated Stochastic Gradient Descent (ASGD) which is a variant of the NAG method and has better performance than the SGD, the NAG and the HB methods.

In our algorithm DP-GDwS, we solve strongly convex problems by using a special type of the HB method with noisy full gradients. The analysis summarized above for the stochastic heavy ball method are not valid for our method because of different assumptions about the algorithm parameters and the objective function selection. The analysis in [88] can be followed for the convergence of DP-GDwS. However, we give a stronger convergence result.

To analyze their algorithms, all of the papers mentioned above use the standard approach which is not easy to construct and differs from one algorithm to another. Recently, a systematic approach based on control theory is preferred to analyze the

first order optimization algorithms [2,3,30,40,48]. These papers use Lyapunov functions which are nonnegative functions representing the current state of an algorithm. After constructing the Lyapunov function, the convergence rate can be found with respect to the rate of decrease of this function. Lessard et. al [48] and Fazlyab et. al [30] use integral quadratic constraints (IQC) approach from robust control theory to construct the Lyapunov function. In [40], it is obtained by using dissipativity which is a notion about energy dissipation in physics. Their approach results in smaller linear matrix inequalities (LMI) which is simpler than the one in [48]. They analyze the NAG method and generalize the approach for various settings. In [2], Aybat et al. analyze the robustness measure to the gradient noise of the GD and the NAG methods. For the quadratic case, they present exact expressions by using robust control theory and tight bounds for smooth strongly convex case by using Lyapunov functions. They also show that the NAG method has a better convergence rate than the GD method under the same robustness. In another study, Aybat et al. [3] introduce a multistage accelerated stochastic algorithm which uses noisy gradients. Their algorithm achieves optimal rate for both deterministic and stochastic settings.

There exist other works presenting accelerated algorithms, for which the design or the analysis is contructed with respect to the dynamic system approach. For example, the analysis of the tradeoff between robustness and performance of the algorithms is considered in [20]. They design a momentum-based algorithm by considering the tradeoff between robustness and performance for smooth strongly convex problems. The rate analysis is done with respect to the approach from control theory as in [48]. In [78], Sun et al. consider the heavy ball problem over convex setting and claim to obtain a better or the same complexity result with the existing studies under weaker assumptions. They obtain the first non-ergodic convergence rate of $O(1/k)$ over co-coercive objective function with constant stepsize again by using the dynamical system approach. The linear convergence of the HB method is also proved under the condition of relaxed strongly convexity which is weaker than strong convexity. The known fastest linear convergence rate for first order algorithms is ρ and ρ^2 for $\rho = 1 - \sqrt{m/L}$ when the objective function is m-strongly convex and the gradient is L-Lipschitz continous. Nesterov's algorithm achieves rate ρ when the function is strongly convex with parameter m and L. In [81], their aim is to present an algorithm which has a better

convergence rate than the globally convergent first order algorithms. They design the Triple Momentum Method by using three momentum terms. In the design of this algorithm, they exploit IQC approach from [48], but the convergence proof does not rely on the approaches from control theory.

In this thesis, we take the advantage of these approaches while analyzing our algorithms DP-GDwS and DP-MAG. For DP-GDwS, we follow the similar steps as in [2]. Their approach is applied on the GD, the NAG methods but not on the HB method. With simple adjustments, we obtain an expression for the convergence rate of our algorithm. For our second deterministic algorithm, DP-MAG, the convergence analysis follows the same steps as in [3].

Chapter 3

An Algorithm Based On Data Distribution

In this chapter, we consider a partially separable objective function whose general form is as in (1.2). There are various strategies for the solution of this type of problems. In this thesis, we focus on a different strategy and instead of classical approach of selecting x_i, we select f_i at each iteration. These approaches are named as *incremental methods*; *cf.* [10]. An incremental method selects a subset from function indices at each iteration, and then takes a step towards the minimum of the selected functions. In other words, for every iteration τ, subset $S^{(\tau)} \subset \mathcal{I}$ is selected and the update is done with respect to the proxy objective function, $\sum_{i \in S^{(\tau)}} f_i(x)$ (see Figure 3.1). Although the actual objective function is never evaluated and different proxy objective is used at each iteration, the incremental algorithms still converge to the solution of the main problem under mild conditions. SGD can be given as an example of this type of algorithms. In our algorithm, we aim to use parallel computation, and hence, we select the subsets carefully such that the proxy objective will be separable.

Next, we provide an example to present the general approach. The rather generic objective function in (1.2) covers many types of optimization problems arising in machine learning. Although we demonstrate our approach with a simple example, the given formulation is compatible with many other problems and it can be easily checked that various problems such as logistic regression, matrix completion can be handled by writing the objective function as partially separable as in Equation (1.2).

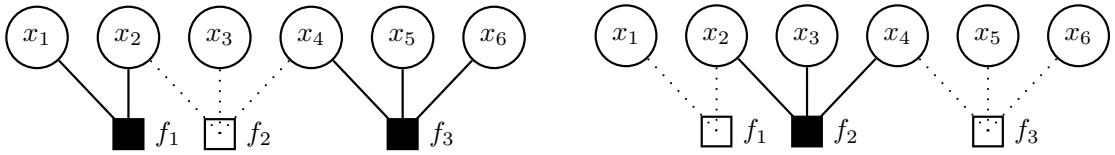


Figure 3.1: Incremental optimization of a partially separable objective by two proxy objectives $f_1(x) + f_3(x)$ and $f_2(x)$. At each step, we pick a subset of functions and ignore the remainings. The black and white squares represent the chosen and omitted functions, respectively. By careful selection of subsets, we can process each proxy objective in parallel and obtain an approximation to the true solution. Based on this scheme, we present a second order algorithm.

Example 3.0.1. [[45], Page 4] *Consider the following matrix factorization problem:*

$$\min_x \left\| \begin{pmatrix} y_1 & y_2 \\ y_3 & y_4 \\ y_5 & y_6 \end{pmatrix} - \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \begin{pmatrix} x_4 & x_5 \end{pmatrix} \right\|_F^2,$$

where $\|\cdot\|_F$ is the Frobenius norm. By using our notation, the objective function can be written as

$$\sum_{i\in\mathcal{I}} f_i(x_{\boldsymbol{\alpha}_i}) = (y_1 - x_1x_4)^2 + (y_2 - x_1x_5)^2 + \cdots + (y_6 - x_3x_5)^2.$$

Clearly, we have $\mathcal{I} = \{1, 2, \ldots, 6\}$ and $\mathcal{J} = \{1, 2, \ldots, 5\}$ with the subsets $\boldsymbol{\alpha}_1 = \{1, 4\}$, $\boldsymbol{\alpha}_2 = \{1, 5\}$, $\boldsymbol{\alpha}_3 = \{2, 4\}$, $\boldsymbol{\alpha}_4 = \{2, 5\}$, $\boldsymbol{\alpha}_5 = \{3, 4\}$, and $\boldsymbol{\alpha}_6 = \{3, 5\}$.

Now, we consider a more general problem which we aim to factorize an observed data matrix $Y \in \mathbb{R}^{K\times N}$. In other words, our aim is to find two factor matrices X_1 and X_2, where $X_1 \in \mathbb{R}^{K\times L}$ and $X_2 \in \mathbb{R}^{L\times N}$ such that

$$\min_{X_1,X_2} \|Y - X_1X_2\|_F^2.$$

In elementwise notation we have

$$f(x) = \sum_{a,b}(y_{a,b} - \sum_k x_{1,a,k}x_{2,k,b})^2. \tag{3.1}$$

Letting $i = (a, b)$, we can write this objective as

$$f(x) = \sum_{i\in\mathcal{I}} f_i(x_{\boldsymbol{\alpha}_i}),$$

where $\boldsymbol{\alpha}_i$ is the set of indices that correspond to the row a of X_1 and column b of X_2. In case some entries of the matrix Y are unknown, the summation in Eq.3.1 is constructed only by using the observed pairs (a, b), which still keeps the form of the final objective.

In this chapter, we make the following contributions:

- We propose a generic second-order algorithm which can be used for large scale convex and nonconvex optimization problems. It has application for the solution of various type of problems such as matrix-tensor factorization, regression, and neural network training.
- The convergence properties of our algorithm for the *deterministic* case is provided. In machine learning, convergence analysis for such a deterministic algorithm has not provided before to the best of our knowledge.
- We also present *stochastic* version which is based on random selection of subsets and demonstrate its convergence analysis.
- By considering several parallelization strategies, we present a simple load balancing heuristic. This approach is especially suitable for parallel solution of matrix and tensor factorization problems since the observed entries is not uniformly distributed.
- With the aim of presenting an application, we give an implementation based on L-BFGS procedure [15].
- In the numerical experiments part, we solve various sized large-scale matrix factorization problems. Our results are compared with a well-known first order method [34] and it is shown that HAMSI achieves faster convergence.

3.1 Deterministic HAMSI

The main idea of our algorithm is iterating over multiple datasets and using of a second order approximation at each iteration. Before derivation of the generic algorithm, we first present an expression for the function index set $\mathcal{I}$. We can write $\mathcal{I}$ as union of mutually disjoint subsets S_k for $k = 1, \ldots, K$ such that

$$\mathcal{I} = S_1 \cup \cdots \cup S_k \cup \cdots \cup S_K.$$

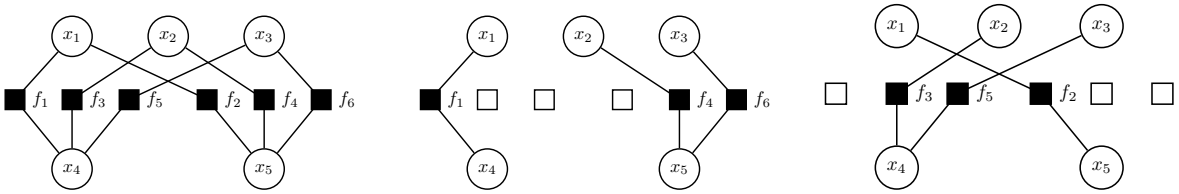


Figure 3.2: (Left) The factor graph of the problem in Example 3.0.1. (Middle and Right) A partitioning of function index set as $\mathcal{I} = \cup_{k=1}^{K} S_k$ where $S_k = S_{k,1} \cup \cdots \cup S_{k,B_k}$ for $k = 1, \ldots, K$. In this example $K = 2$ and with number of blocks $B_1 = B_2 = 2$. (Middle) First subset ($k = 1$) with $S_1 = S_{1,1} \cup S_{1,2} = \{1\} \cup \{4, 6\}$ and $\boldsymbol{\alpha}_{1,1} = \{1, 4\}$, $\boldsymbol{\alpha}_{1,2} = \{2, 3, 5\}$, (Right) Second subset ($k = 2$) with $S_2 = S_{2,1} \cup S_{2,2} = \{3, 5\} \cup \{2\}$, $\boldsymbol{\alpha}_{k,1} = \{2, 3, 4\}$ and $\boldsymbol{\alpha}_{k,2} = \{1, 5\}$.

This collection of subsets is referred as a *cover* and denoted by $\mathcal{S} \equiv \{S_1, \ldots, S_K\}$. We partition $\mathcal{I}$ into subsets, moreover the subsets $S_k \in \mathcal{S}$ for $k = 1, \ldots, K$ can be further partitioned into mutually exclusive blocks B_k as

$$S_k = S_{k,1} \cup \cdots \cup S_{k,b} \cup \cdots \cup S_{k,B_k}.$$

For the example in Figure 3.1, the function index set has the following partition which we first write as union of subsets and then as union of blocks:

$$\mathcal{I} = S_1 \cup S_2 = (S_{1,1} \cup S_{1,2}) \cup S_{2,1} = (\{1\} \cup \{3\}) \cup \{2\}.$$

There are various ways for chosing the cover $\mathcal{S}$. After fixing the cover, the partition of S_k can be decided by using the factor graph. There, the individual blocks can be optimized independently. To achieve that, the partition of subsets should be done carefully and the variables from each function f_i such that $i \in S_{k,b}$ should be mutually disjoint where $S_{k,b}$ represents the bth block of the subset at k-th iteration. To increase the degree of paralellism, the number of blocks can be increased since the parallelism in a subset S_k is limited by the number of blocks in this subset.

To clarify the partition and make it easy to follow, we use a bipartite graph. The partitioning framework for our simple problem in Example 3.0.1 is given in Figure 3.2. In this example, we aim to solve a small matrix factorization problem for which some entries are missing and we complete these entries by using our algorithm.

Formally, we can write the function index set as union of subsets and related blocks

as follows

$$\mathcal{I} = \bigcup_{k=1}^{K} \bigcup_{b=1}^{B_k} S_{k,b}.$$

Thus, the partially separable objective function whose generic form is given in (1.2) can be written as

$$\min_{x \in \mathbb{R}^{|\mathcal{J}|}} \sum_{k=1}^{K} \sum_{b=1}^{B_k} \sum_{i \in S_{k,b}} f_i(x_{\boldsymbol{\alpha}_i}). \tag{3.2}$$

The parallelization of this problem in (3.2) is possible because of its separability over the second summation which is indexed by b. In other words, we use the mutually disjoint nature of the parameter sets at each block of a subset. To achieve this we define

$$\boldsymbol{\alpha}_{k,b} \equiv \bigcup_{i \in S_{k,b}} \boldsymbol{\alpha}_i \quad \text{for all } k = 1, \ldots, K; b = 1, \ldots, B_k,$$

and require $\boldsymbol{\alpha}_{k,b} \cap \boldsymbol{\alpha}_{k,b'} = \emptyset$ for $b \neq b'$ and for all $k = 1, \ldots, K$. We need the equality $\boldsymbol{\alpha}_{k,b} \cap \boldsymbol{\alpha}_{k,b'} = \emptyset$ for the parallel and exact computation of the (partial) gradients. On the other hand, there exists synchronization-free algorithms in literature which the parameter sets in blocks may overlap. We give further explanation about this case in Section 3.5. In any case we have,

$$f_{k,b}(x_{\boldsymbol{\alpha}_{k,b}}) = \sum_{i \in S_{k,b}} f_i(x_{\alpha_i}).$$

Now, we proceed to present the final form of our optimization problem:

$$\min_{x \in \mathbb{R}^{|\mathcal{J}|}} \sum_{k=1}^{K} \sum_{b=1}^{B_k} f_{k,b}(x_{\boldsymbol{\alpha}_{k,b}}). \tag{3.3}$$

The proposed algorithm uses incremental gradients and a second order information which comes from an approximation to the Hessian of the objective function. Because of the usage of multiple subsets in addition to incremental gradients and second order information, our algorithm is called Hessian Approximated Multiple Subsets Iteration (HAMSI).

The main idea of the algorithm is using of local *convex* quadratic approximation while computing the step. The local quadratic approximation can be expressed as

$$Q(z; \hat{x}, g, H, \beta) \equiv (z - \hat{x})^\top g + \frac{1}{2}(z - \hat{x})^\top H (z - \hat{x}) + \frac{1}{2}\beta \|z - \hat{x}\|^2, \tag{3.4}$$

Algorithm 1: Hessian Approximated Multiple Subsets Iteration (HAMSI)

1 **input:** x, H, $\boldsymbol{\alpha}_{k,b}$ for all $k = 1, \ldots, K$; $b = 1, \ldots, B_k$.
2 $t = 1$
3 $\mathcal{K} = [K, 1, 2, \ldots, K-1, K]$
4 **repeat**
5 Update β_t (*increasing sequence*)
6 $\mathcal{K} \leftarrow$ SetSchedule($\mathcal{K}$)
7 **for** $\ell = 1, 2, \ldots, K+1$ **do**
8 $k = \mathcal{K}[\ell]$
9 **for** $b = 1, 2, \ldots, B_k$ **do in parallel**
10 $x_{\boldsymbol{\alpha}_{k,b}} = \arg\min_z Q(z; x_{\boldsymbol{\alpha}_{k,b}}, \nabla f_{k,b}(x_{\boldsymbol{\alpha}_{k,b}}), H_{\boldsymbol{\alpha}_{k,b}}, \beta_t)$
11 Evaluate approximate Hessian matrix H at x
12 $t \leftarrow t + 1$
13 **until** *convergence or* $t >$ *max_ epochs*

where g is an incremental gradient, H is (an approximation to) the Hessian of the objective function. The parameter β is important to bound the stepsize and control the oscillation of the incremental steps.

The high-level pseudocode of HAMSI is given in Algorithm 1. In the algorithm, x represents the current solution. At each *inner iteration* k, the functions in related subset S_k (lines 7-10) are used. The selection of subsets is deterministic and the order of the selection is determined by a SetSchedule function. SetSchedule takes the current order as parameter and obtain the new order by applying one cyclic left-shift to the first K elements of $\mathcal{K}$. The first and the last element of the sequence becomes the same number. To explain the framework clearly, we give an illustrative example for $K = 5$:

$$\begin{aligned}
&\text{Iteration } t && [3,4,5,1,2,3] = \text{SetSchedule}([2,3,4,5,1,2]) \\
&\text{Iteration } t+1 && [4,5,1,2,3,4] = \text{SetSchedule}([3,4,5,1,2,3]) \\
&\text{Iteration } t+2 && [5,1,2,3,4,5] = \text{SetSchedule}([4,5,1,2,3,4]) \\
&\vdots && \vdots
\end{aligned}$$

In this setting, the subsets are defined once at the beginning and SetSchedule function only reorders a schedule array $\mathcal{K}$ of size $K+1$ appropriately. As it is mentioned,

we use the same number as the first and last element in $\mathcal{K}$. This approach is preferred since updating the Hessian approximation with the same subsets is reasonable for practical purposes. The same rule is also applied for the example implementation given in Algorithm 3 in Section 3.4. We note that this setting does not change the convergence analysis.

For each subset, the corresponding blocks are traversed (lines 9 and 10) and the corresponding part of the solution is updated. The updated part can be denoted as $\bigcup_{b=1}^{B_k} \boldsymbol{\alpha}_{k,b}$.

An important point is that the blocks of the inner step are used in parallel in inner loop and we use different blocks from the same approximate Hessian matrix H at all inner iterations until completion of the t^{th} cycle. In other words, for each block b from subset k, the related submatrix of H is denoted by $H_{\boldsymbol{\alpha}_{k,b}}$, where $H_{\boldsymbol{\alpha}} = \{H(i,i') : i,i' \in \boldsymbol{\alpha}\}$. Here, β is a constant parameter during inner iterations and it is updated for each new iteration as shown in line 5. We further explain the update rule for β at the end of convergence analysis.

There exists some details about the implementation of the algorithm which can be seen from the description above. To exemplify these details, which are related to the construction of quadratic approximation (line 10) and the solution of the corresponding subproblem, we provide an explicit implementation in Section 3.4.

Next, we show the convergence properties of HAMSI when the order of subset selection is deterministic. The construction of HAMSI is similar to the one given in [52, 75]. In the previous analysis of HAMSI, the convergence is showed by proving the limit of the infimum of the gradient converges to zero. Here, we provide an improved convergence analysis showing the limit of the gradient tends to zero as the number of iterations increases.

To simplify our notation, we define

$$\nabla f_{S_k}(x) \equiv \sum_{b=1}^{B_k} \nabla f_{k,b}(x_{\alpha_{k,b}}).$$

The solution and approximate Hessian at iteration t is denoted by by $x^{(t)}$ and H^t, respectively. We update the current solution in lines 7-10. Those inner iterates are denoted by $x^{(t,\ell)}$ and the very last inner iterate $x^{(t,K+1)}$ becomes $x^{(t+1)}$ after line 10.

We first list the assumptions before presenting our analysis:

A.1 The twice differentiable objective function f is bounded below.

A.2 The Hessian matrices for the component functions are uniformly bounded at every iteration. That is, for every S_k and t, we have

$$\|\nabla^2 f_{S_k}(x^{(t)})\| \leq L,$$

where L is the well-known Lipschitz constant.

A.3 The eigenvalues of the approximation matrices are bounded so that

$$(U + \beta_t)^{-1} = U_t \leq \|(H^t + I\beta_t)^{-1}\| \leq M_t = (M + \beta_t)^{-1}$$

holds. Here, U_t and M_t are known constants with $0 < M \leq U$, and I denotes the identity matrix.

A.4 The gradient norms are uniformly bounded at every iteration t. That is, for every S_k we have

$$\|\nabla f_{S_k}(x^{(t)})\| \leq C,$$

where C is a known constant.

The proof of main convergence theorem is completed by using two lemmas. The first lemma helps us to present a bound on the difference between the true gradient of a block at $x^{(t)}$ and the evaluated gradient at the inner iterate $x^{(t,\ell)}$. The result obtained from the second lemma is related to the bound on the error committed by taking incremental steps at the inner iterates $x^{(t,\ell)}$ instead of the exact Newton step at $x^{(t)}$. Then, we present our convergence result with our main theorem.

Note that we use $S_{[\ell]}$ instead of $S_{\mathcal{K}[\ell]}$ to underline the dependence of subset selection on the index ℓ. Moreover, we define $x^{(t,0)} \equiv x^{(t)}$ to clarify our summations.

Lemma 3.1.1. *In Algorithm 1, let $x^{(t)}$ be the solution at iteration t and $x^{(t,\ell)}$ be the inner iteration ℓ. Then, we have*

$$\|\nabla f_{S_\ell}(x^{(t,\ell-1)}) - \nabla f_{S_\ell}(x^{(t)})\| \leq LM_tC(\ell - 1). \tag{3.5}$$

Proof. By using Assumption A.2, we have

$$\begin{aligned}
\|\nabla f_{S_\ell}(x^{(t,\ell-1)}) - \nabla f_{S_\ell}(x^{(t)})\| &\leq L\|x^{t,l-1} - x^{(t)}\| \\
&= L\|x^{t,l-1} - x^{t,l-2} + x^{t,l-2} - x^{t,l-3} + \cdots + x^{t,1} - x^{(t)}\| \\
&\leq L\textstyle\sum_{j=1}^{\ell-1} \|x^{t,j} - x^{t,j-1}\|.
\end{aligned}$$

Note for $j = 1, \cdots, \ell - 1$ that

$$\|x^{t,j} - x^{t,j-1}\| = \|x^{t,j-1} - (H^t + \beta_t I)^{-1} \nabla f_{S_j}(x^{(t,j-1)}) - x^{t,j-1}\| \leq M_t C,$$

where the last inequality holds by Assumption A.4. Therefore, we have

$$\|\nabla f_{S_\ell}(x^{(t,\ell-1)}) - \nabla f_{S_\ell}(x^{(t)})\| \leq L M_t C(\ell - 1).$$

□

We obtain a bound on the difference between two consecutive iterations with the next lemma.

Lemma 3.1.2. *Given two consecutive iterations, t and $t+1$ of Algorithm 1, we have*

$$\|x^{(t+1)} - x^{(t)}\| \leq \frac{B + C(M+1)}{M+1} M_t,$$

where $B \equiv \frac{1}{2} LCK(K+1)$.

Proof. At iteration $t+1$, we have

$$\begin{aligned} x^{(t+1)} &= x^{(t)} - \textstyle\sum_{\ell=1}^{K+1} (H^t + \beta_t I)^{-1} \nabla f_{S_\ell}(x^{(t,\ell-1)}) \\ &= x^{(t)} - (H^t + \beta_t I)^{-1} \nabla f(x^{(t)}) \\ &\quad + (H^t + \beta_t I)^{-1} \textstyle\sum_{\ell=1}^{K+1} (\nabla f_{S_\ell}(x^{(t)}) - \nabla f_{S_\ell}(x^{(t,\ell-1)})). \end{aligned}$$

This shows that

$$x^{(t+1)} - x^{(t)} = \Delta_t - (H^t + \beta_t I)^{-1} \nabla f(x^{(t)}), \tag{3.6}$$

where

$$\Delta_t \equiv (H^t + \beta_t I)^{-1} \sum_{\ell=1}^{K+1} (\nabla f_{S_\ell}(x^{(t)}) - \nabla f_{S_\ell}(x^{(t,\ell-1)})).$$

Using now (3.5) implies

$$\begin{aligned} \|\Delta_t\| &\leq M_t \textstyle\sum_{\ell=1}^{K+1} \|\nabla f_{S_\ell}(x^{(t)}) - \nabla f_{S_\ell}(x^{(t,\ell-1)})\| \\ &\leq M_t \textstyle\sum_{\ell=1}^{K+1} L M_t C(\ell - 1) = \frac{1}{2} L M_t^2 CK(K+1) = B M_t^2. \end{aligned} \tag{3.7}$$

Then, we obtain

$$\begin{aligned} \|x^{(t+1)} - x^{(t)}\| &= \|\Delta_t - (H^t + \beta_t I)^{-1} \nabla f(x^{(t)})\| \\ &\leq \|\Delta_t\| + \|(H^t + \beta_t I)^{-1}\| \nabla f(x^{(t)})\| \\ &\leq B M_t^2 + C M_t \leq \tfrac{B + C(M+1)}{M+1} M_t. \end{aligned}$$

□

Finally,the convergence of our algorithm is presented in the following theorem.

Theorem 3.1.3. *Consider the iterates $x^{(t)}$ of Algorithm 1. Suppose that $\beta^{(t)} \geq 1$ is chosen to satisfy*

$$\sum_{t=1}^{\infty} U_t = \infty \text{ and } \sum_{t=1}^{\infty} M_t^2 < \infty. \tag{3.8}$$

Then,

$$\lim_{t\uparrow\infty} \nabla f(x^{(t)}) = 0,$$

and for each accumulation point x^* of the sequence $\{x^{(t)}\}$, we have $\nabla f(x^*) = 0$.

Proof. As f is a twice differentiable function, we have

$$f(x^{(t+1)}) - f(x^{(t)}) \leq \nabla f(x^{(t)})^T (x^{(t+1)} - x^{(t)}) + \frac{LK}{2} \|x^{(t+1)} - x^{(t)}\|^2.$$

Using now Lemma 3.1.2 along with (3.6) and (3.7), we obtain

$$\begin{aligned} f(x^{(t+1)}) - f(x^{(t)}) &\leq \nabla f(x^{(t)})^T \Delta_t - \nabla f(x^{(t)})^T (H^t + \beta_t I)^{-1} \nabla f(x^{(t)}) + \tfrac{LK}{2} \|x^{(t+1)} - x^{(t)}\|^2 \\ &\leq \|\nabla f(x^{(t)})\| \|\Delta_t\| - U_t \|\nabla f(x^{(t)})\|^2 + \tfrac{LK}{2} \left(\tfrac{B+C(M+1)}{M+1}\right)^2 M_t^2 \\ &\leq -U_t \|\nabla f(x^{(t)})\|^2 + \bar{B} M_t^2, \end{aligned} \tag{3.9}$$

where $\bar{B} \equiv CB + \frac{LK}{2} \left(\frac{B+C(M+1)}{M+1}\right)^2$. Due to Assumption A.1, we can write $\inf_{x\in\mathbb{R}^n} f(x) = f^* > -\infty$. Thus, we obtain

$$0 \leq f(x^{(t+1)}) - f^* \leq f(x^{(t)}) - f^* + \bar{B} M_t^2.$$

Relation (3.8) and Lemma 2.2 in [52] together show that the sequence $\{f(x^{(t)})\}$ converges. By using (3.9), we further have

$$\begin{aligned} f(x^{(1)}) - f^* &\geq f(x^{(1)}) - f(x^{(t)}) = \textstyle\sum_{j=1}^{t-1} (f(x^{(t)}) - f(x^{(t+1)})) \\ &\geq \textstyle\sum_{j=1}^{t-1} U_j \|\nabla f(x^{(j)})\|^2 - \bar{B} \sum_{j=1}^{t-1} M_j^2 \\ &\geq \inf_{1\leq j\leq t-1} \|\nabla f(x^{(j)})\|^2 \textstyle\sum_{j=1}^{t-1} U_j - \bar{B} \sum_{j=1}^{t-1} M_j^2. \end{aligned}$$

Let now $t \to \infty$, then

$$f(x^{(1)}) - f^* \geq \inf_{j\geq 1} \|\nabla f(x^{(j)})\|^2 \sum_{j=1}^{\infty} U_j - \bar{B} \sum_{j=1}^{\infty} M_j^2. \tag{3.10}$$

Using again Assumption A.1 and conditon (3.8), we obtain

$$\inf_{t\geq 1} \|\nabla f(x^{(t)})\| = 0. \tag{3.11}$$

Now, suppose for contradiction that the sequence $\{\nabla f(x^{(t)})\}$ does not converge to zero. Then, there exists an increasing sequence of integers such that for some $\varepsilon > 0$, we have $\|\nabla f(x^{(t_\tau)})\| \geq \varepsilon$ for all τ. On the other hand, the relation (3.11) implies that there exist some $j > t_\tau$ such that $\|\nabla f(x^{(j)})\| \leq \frac{\varepsilon}{2}$. Let j_τ be the least integer for each τ satisfying these inequalities. Then, we have

$$\begin{aligned}\tfrac{\varepsilon}{2} &\leq \|\nabla f(x^{(t_\tau)})\| - \|\nabla f(x^{(j_\tau)})\| \\ &\leq \|\nabla f(x^{(t_\tau)}) - \nabla f(x^{(j_\tau)})\| \\ &\leq LK\|x^{t_\tau} - x^{j_\tau}\| \leq LK\tfrac{B+C(M+1)}{M+1}\textstyle\sum_{k=t_\tau}^{j_\tau-1} M_k,\end{aligned}$$

where the last inequality follows from Lemma 3.1.2. Since $0 < M \leq U$, there exists $\zeta \leq \frac{M}{U} \leq 1$ such that

$$M + \beta_k \geq \zeta U + \beta_k \geq \zeta(U + \beta_k) \implies M_k \leq \frac{1}{\zeta} U_k.$$

Thus, we obtain

$$0 < \hat{B} \equiv \frac{\varepsilon(M+1)\zeta}{2LK(B + C(M+1))} \leq \sum_{k=t_\tau}^{j_\tau-1} U_k.$$

Then, using together with inequality (3.9), we obtain

$$\begin{aligned}f(x^{(t_\tau)}) - f(x^{(j_\tau)}) &\geq \textstyle\sum_{k=t_\tau}^{j_\tau-1} U_k\|\nabla f(x^{(k)})\|^2 - \bar{B}\sum_{k=t_\tau}^{j_\tau-1} M_k^2 \\ &\geq \hat{B}\inf_{t_\tau \leq k \leq j_\tau - 1}\|\nabla f(x^{(k)})\|^2 - \bar{B}\textstyle\sum_{k=t_\tau}^{\infty} M_k^2.\end{aligned}$$

Since the left-hand-side of the inequality converges and the condition (3.8) holds, we have

$$\lim_{\tau\uparrow\infty}\inf_{t_\tau \leq k \leq j_\tau - 1}\|\nabla f(x^{(k)})\|^2 = 0. \tag{3.12}$$

But our choice of t_τ and j_τ guarantees $\|\nabla f(x^{(k)})\| > \frac{\varepsilon}{2}$ for all $t_\tau \leq k \leq j_\tau$, and hence, we arrive at a contradiction with (3.12). Therefore, $\nabla f(x^{(t)})$ converges, and with the continuity of the gradient, we conclude for each accumulation point x^* of the sequence $\{x^{(t)}\}$ that $\nabla f(x^*) = 0$ holds. □

Clearly, the simple choice of $\beta^{(t)} = (\eta t)^\gamma$ with $\eta > 0$ and $\gamma \in (0.5, 1]$ satisfies condition (3.8).

Algorithm 2: Stochastic HAMSI

input: x, H, $\boldsymbol{\alpha}_{k,b}$ for all $k = 1, \cdots, K$; $b = 1, \cdots, B_k$.
$t = 1$
$\mathcal{K} = [K, 1, 2, \cdots, K-1, K]$
repeat
  Update β_t (*increasing sequence*)
  $\mathcal{K} \leftarrow$ SHUFFLE($\mathcal{K}$)
  for $\ell = 1, 2, \cdots, K+1$ **do**
    $k = \mathcal{K}[\ell]$
    for $b = 1, 2, \cdots, B_k$ **do in parallel**
      $x_{\boldsymbol{\alpha}_{k,b}} = \arg\min_z Q(z; x_{\boldsymbol{\alpha}_{k,b}}, \nabla f_{k,b}(x_{\boldsymbol{\alpha}_{k,b}}), H_{\boldsymbol{\alpha}_{k,b}}, \beta_t)$
  Evaluate approximate Hessian matrix H at x
  $t \leftarrow t + 1$
until *convergence or* $t >$ *max_ epochs*

3.2 Stochastic HAMSI

We next propose a stochatic version of HAMSI which is based on the random selection of the blocks instead of following a deterministic schedule as shown in the previous section. The pseudocode of this version is given in Algorithm 2. The only difference with HAMSI is the random selection of the subsets of from data. The analysis of stochastic HAMSI is presented for the selection of subsets with replacement. However, we provide the numerical experiments for without replacement scheme, which corresponds to "shuffling". This approach has been used in the literature before [34].

Now, we proceed to prove the convergence of the proposed algorithm HAMSI when the order of subset selection is random. The idea of the presented proof comes from the analysis given in [12]. During this part we will use $\hat{H}^{(t)} = (H^{(t)} + \beta^{(t)} I)^{-1}$, which is second order information at iteration t, to ease the notation.

Our discussion is given under the some assumptions.

A.1 The objective function is three times differentiable with continuous derivatives and it is bounded from below.

A.2 The Hessian matrices for the component functions are uniformly bounded at every iteration t. That is, for every S_k and t, we have

$$||\nabla^2 f_{S_k}|| \leq L,$$

where L is the Lipschitz constant.

A.3 The eigenvalues of the approximation matrices are bounded so that

$$(U + \beta_t)^{-1} = U_t \leq \|(H^t + I\beta_t)^{-1}\| \leq M_t = (M + \beta_t)^{-1}$$

holds. Here, U_t and M_t are known constants with $0 < M \leq U$, and I denotes the identity matrix.

A.4 The gradient norms are uniformly bounded at every iteration t; i.e., for every S_k we have

$$||\nabla f_{S_k}(x^{(t)})|| \leq C,$$

where C is a constant.

A.5 When the norm of the parameter x is larger than a certain horizon D, the opposite of the gradient $-\nabla f(x)$ points towards the origin.

$$inf_{x^2 \geq D} x^T \nabla f(x) > 0.$$

Before presenting the main theorem, we prove that the iterates x_t are confined in a bounded region by using Assumption A.5.

Lemma 3.2.1. *The parameter vectors $x^{(t)}$ are confined almost surely in a bounded region of the parameter space.*

Proof. We give the proof in three steps.

Step a. We define a suitable criterion

$$u_t = \rho(||x^{(t)}||^2)$$

with

$$\rho(x) \triangleq \begin{cases} 0 & if x < D \\ (x - D)^2 & if x \geq D. \end{cases}$$

Step b. By using the definition of ρ, the following expression is satisfied

$$\rho(y) - \rho(x) \leq (y - x)\rho'(x) + (y - x)^2. \tag{3.13}$$

The inequality in (3.13) becomes equality when $x > D$ and $y > D$. Next, we apply (3.13) to the difference $u_{t+1} - u_t$,

$$u_{t+1} - u_t = \rho(||x^{(t+1)}||^2) - \rho(||x^{(t)}||^2).$$

After necessary arrangements and taking expectation over past information $\mathcal{P}_t$ we obtain

$$\begin{aligned} E\left[u_{t+1} - u_t | \mathcal{P}_t\right] \leq -2(x^{(t)})^T \hat{H}^{(t)} \nabla f(x^{(t)}) \rho'\left(||x^{(t)}||^2\right) + M_t^2 C^2 \rho'\left(||x^{(t)}||^2\right) \\ +4||x^{(t)}||^2 M_t^2 C^2 - 4||x^{(t)}|| M_t^3 C^3 + M_t^4 C^4, \end{aligned} \tag{3.14}$$

$$E\left[u_{t+1} - u_t | \mathcal{P}_t\right] \leq -2(x^{(t)})^T \hat{H}^{(t)} \nabla f(x^{(t)}) \rho'\left(||x^{(t)}||^2\right) + M_t^2 C^2 A. \tag{3.15}$$

When $||x^{(t)}||^2 < D$, the first term of the right hand side of this inequality is null because $\rho'\left(||x^{(t)}||^2\right) = 0$. When $||x^{(t)}||^2 \geq D$, the first term of the right hand side of this inequality is negative because of Assumption A.5. Thus,

$$E\left[u_{t+1} - u_t | \mathcal{P}_t\right] \leq M_t^2 C^2 A,$$

where A is a positive constant.

Step c. Assume that u_t converges a value u_∞ greater than 0. When the parameter t is large enough, the convergence of u_t implies that $\left\|x^{(t)}\right\|^2 > D$ and $\left\|x^{(t+1)}\right\|^2 > D$. As it is mentioned earlier, the inequality (3.13) becomes equality and it implies that the infinite sum given below converges almost surely:

$$\sum_{t=1}^{\infty} (x^{(t)})^T \hat{H}^{(t)} \nabla f(x^{(t)}) \rho'\left((x^{(t)})^2\right) < \infty. \tag{3.16}$$

This result is not compatible with Assumption A.5. We must therefore conclude that u_t converges to zero.

□

The convergence of u_t requires that the norm u_t^2 is bounded, and thus, the confinement of parameters in a bounded region is proved with the help of Assumption A.5. Since the confinement property is satisfied, it is guaranteed that all continuous functions of $x^{(t)}$, such as $||x^{(t)}||^2$, $\nabla f(x^{(t)})$ and, all derivatives of f, are bounded. Thus, we can introduce positive constants K_1, K_2, ... whenever we need a bound for such expressions in the rest of analysis.

Theorem 3.2.2. *The function $f(x^{(t)})$ converges almost surely and the gradient $\nabla f(x^{(t)})$ converges to 0 almost surely.*

Proof. The proof is again completed in three steps.

Step a. The first step is again to define a suitable criterion function,

$$f_t = f(x^{(t)}) \geq 0.$$

Step b. The variations of f_t can be bounded by using a first order Taylor expansion and bounding the second order derivatives with L with the help of Assumption A.2:

$$\left| f_{t+1} - f_t + \left[\hat{H}^{(t)} \nabla f_{S_t}(x^{(t)})\right]^T \nabla f(x^{(t)}) \right| \leq ||\hat{H}^{(t)} \nabla f_{S_t}(x^{(t)})||^2 L$$

and thus,

$$f_{t+1} - f_t \leq -\left[\hat{H}^{(t)} \nabla f_{S_t}(x^{(t)})\right]^T \nabla f(x^{(t)}) + ||\hat{H}^{(t)} \nabla f_{S_t}(x^{(t)})||^2 L.$$

Now, we take the conditional expectation and obtain

$$E[f_{t+1} - f_t|\mathcal{P}_t] \leq -\left[\hat{H}^{(t)} \nabla f(x^{(t)})\right]^T \nabla f(x^{(t)}) + ||\hat{H}^{(t)}||^2 E\left[||\nabla f_{S_t}(x^{(t)})||^2 |\mathcal{P}_t\right] L. \tag{3.17}$$

Since $\hat{H}^{(t)}$ is a positive definite matrix, we have

$$E[f_{t+1} - f_t|\mathcal{P}_t] \leq ||\hat{H}^{(t)}||^2 E\left[||\nabla f_{S_t}(x^{(t)})||^2 |\mathcal{P}_t\right] L,$$

and

$$[f_{t+1} - f_t|\mathcal{P}_t] \leq M_t^2 \frac{1}{K} \sum_{t=1}^{K} ||f_{S_t}(x^{(t)})||^2 L,$$

$$E[f_{t+1} - f_t|\mathcal{P}_t] \leq M_t^2 C^2 K_1.$$

Then, we can bound the positive expected variations of h_t as follows

$$E[\delta_t(f_{t+1} - f_t)] = E[\delta_t E[f_{t+1} - f_t|\mathcal{P}_t]] \leq M_t^2 C^2 L.$$

This bound is the summation of convergent infinite sums, and the Quasi-Martingale Theorem [12] implies that $f_t = f(x^{(t)})$ converges almost surely,

$$f(x^{(t)}) \to f_\infty.$$

Step c. In this last step, we prove that the gradient vector $\nabla f(x^{(t)})$ converges to zero almost surely. By taking expectation of (3.17) and summing on $t = 1, ..., \infty$, we obtain

$$\sum_{t=1}^{\infty} E[f_{t+1} - f_t | \mathcal{P}_t] \leq -\sum_{t=1}^{\infty} \left[\hat{H}^{(t)} \nabla f(x^{(t)})\right]^T \nabla f(x^{(t)}) + \sum_{t=1}^{\infty} M_t^2 C^2 L. \quad (3.18)$$

We see that the convergence of $f(x^{(t)})$ implies the convergence of the following infinite sum:

$$\sum_{t=1}^{\infty} \left[\hat{H}^{(t)} \nabla f(x^{(t)})\right]^T \nabla f(x^{(t)}) < \infty. \quad (3.19)$$

The convergence of the squared gradient $\nabla f(x^{(t)})$ is not clear yet. We define a second criterion

$$g_t = \left(\nabla f(x^{(t)})\right)^2.$$

By using Taylor expansion procedure for f_t, we can bound the variations of g_t.

$$g_{t+1} - g_t \leq -2 \left[\hat{H}^{(t)} \nabla f_{S_t}(x^{(t)})\right]^T \nabla^2 f(x^{(t)}) \nabla f(x^{(t)}) + \left[\hat{H}^{(t)} \nabla f_{S_t}(x^{(t)})\right]^2 K_1.$$

Now, we take conditional expectation and use K_4 as bound for the second derivatives. The resulting inequality is given by

$$E[g_{t+1} - g_t | \mathcal{P}_t] \leq 2 \left[\hat{H}^{(t)} \nabla f(x^{(t)})\right]^T \nabla f(x^{(t)}) K_4 + M_t^2 C^2 K_1. \quad (3.20)$$

We can also bound the positive expected variations of g_t as

$$E[\delta_t (g_{t+1} - g_t)] = E[\delta_t E(g_{t+1} - g_t | \mathcal{P}_t)] \leq 2 \left[\hat{H}^{(t)} \nabla f(x^{(t)})\right]^T \nabla f(x^{(t)}) K_2 + M_t^2 C^2 K_1. \quad (3.21)$$

The two terms on the right hand side are the summands of convergent infinite sums. The Quasi-Martingale Theorem [12] implies that g_t converges almost surely. We know that

$$\sum_{t=1}^{\infty} \left[\hat{H}^{(t)} \nabla f(x^{(t)})\right]^T \nabla f(x^{(t)}) < \infty. \quad (3.22)$$

Thus, $g_t \to 0$ as $t \to \infty$ and $\nabla f(x^{(t)}) \to 0$ as $t \to \infty$.

□

3.3 Partitioning And Parallelization

To achieve our second aim of improving the performance of HAMSI, we investigate some parallelization schemes which can be applied to gradient-based optimization algorithms. Designing parallel optimization algorithms is based on finding a suitable cover $\mathcal{S} = \{S_1, \ldots, S_K\}$ for function index set $\mathcal{I}$ and partitioning each subset $\mathcal{S}$ into blocks. In numerical experiments, we describe three parallelization strategy COLOR, HOGWILD, and STRATA and present their variants COLOR-B and STRATA-B, which are suitable for partially separable objectives as in our objective (3.3).

The first parallelization method COLOR is based on coloring of the bipartite graph representation of the problem. For a given bipartite graph $\mathcal{G} = (\mathcal{I}, \mathcal{J}, \mathcal{E})$, the coloring is done in the following basis: if any two function vertices in $\mathcal{I}$ are adjacent to at least one parameter vertices in $\mathcal{J}$, we color them differently. If this is satisfied for all vertices, then the coloring is called *valid*. After obtaining a valid coloring, the function vertices $\mathcal{I}$ can be partitioned into disjoint subsets and each block in the same subset can be processed in parallel.

To apply coloring-based parallelization, there are two necessary conditions: (i) the number of synchronization points should be as small as possible; (ii) the load distribution among the processors should be balanced. The formal definition of graph coloring problem requires minimum number of colors and thus the first condition is satisfied. Although keeping the number of colors small is not easy, there exists some suitable, cheap heuristics that can be used for bipartite-graph coloring [33]. For HAMSI, we will use a heuristic which is based on visiting the vertices in some order and using the suitable color with the smallest index (*first-fit policy*) [33].

After completing the coloring, since the subsets are well defined, we can distribute the blocks in any subset evenly to the processors. However, we have observed from the first experiments that first-fit policy results in an unbalanced distribution of functions. Although the smaller numbered color sets has very large cardinality, the color sets with larger indexes have a few functions. To solve this problem, we present a simple heuristic; color selection is done randomly instead of taking the one with smaller index. Additionally, random selection is firstly done among the available colors used before, if it is not possible, a new color is selected. By using this approach, COLOR-B, a balanced distribution of functions into color sets is achieved as it can be seen from Figure 3.3.

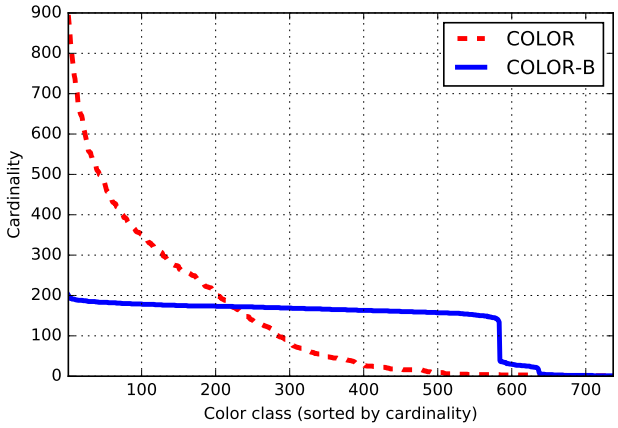
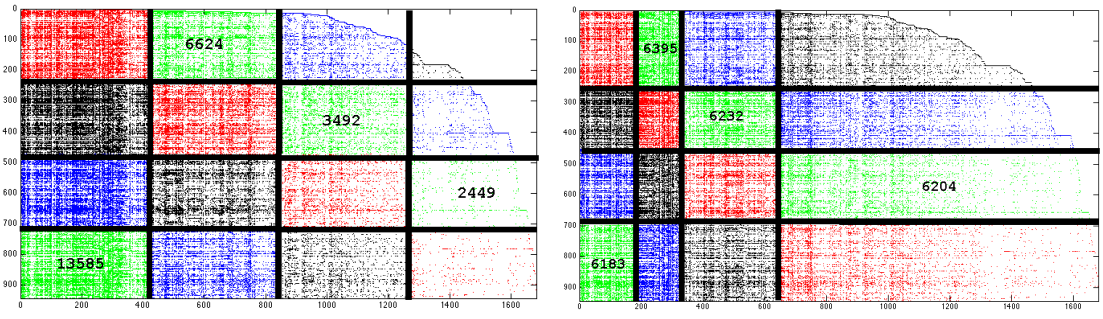


Figure 3.3: The number of functions in each color set for MovieLens 100K dataset. (COLOR) uses a first-fit approach and (COLOR-B) selects a random available color.

As it is obtained form our experiments, the minimum number of colors can be in the order of thousands for a valid coloring. Thus, we cannot assign a small number for K, the number of subsets. To fix this issue for both COLOR and COLOR-B, we firstly package the colors into K bins such that each bin obtains almost equal number of functions. Packaging is done with a first-fit policy and the new color is put into the bin including has less than $|\mathcal{I}|/K$ functions. Then the k-th bin is taken as S_k, the k-th subset.

The second parallelization technique we employ is HOGWILD. The idea is to distribute the functions into K subsets randomly such that each subset has almost the same number of functions. Again, similar to COLOR, each block includes a single function. The disadvantage of this method is that the parameter sets in the blocks may overlap and this causes incorrect calculations of some gradient entries in parallel processing. However, HOGWILD assumes that even there exists error in the gradient, it is not significant because of the large sized parameter sets.

The last parallelization method, *stratification*, is recently employed on parallelization of SGD on distributed setting in [34]. For the problems in multi-dimensional form such as matrix factorization, this method is based on partitioning the parameter dimensions into intervals. Thus, a number of function strata is obtained and they can be processed in parallel. In Figure 3.4, we give two stratifications example for MovieLens 100K matrix. The rows and columns in the matrix represent 943 users and 1682



(a) STRATA (b) STRATA-B

Figure 3.4: Two stratifications for the MovieLens 100K matrix (Left) STRATA uses equi-length intervals. (Right) STRATA-B uses a balancing scheme with non-equal intervals. The number of functions included is written on each rectangle.

movies, respectively. The subsets S_k for $k \in \{1, 2, 3, 4\}$ are shown with stratums and the blocks $B_k = 4$ for $k \in \{1, 2, 3, 4\}$ are represented by colored rectangles. Here, the functions in different strata are given in different colors. The block intervals are different in each subset, thus, we can process the blocks in the same subset in parallel. Although the original framework used in [34] does not aim to balance the number of functions in the blocks of the same subset, there exists studies considering equi-length intervals in the literature. We call this version with equi-length interval as STRATA.

In Figure 3.4(a), STRATA is employed, but it is clearly seen that the function-to-block distribution is still unbalanced because of sparsity and irregularity of the data. To fix this load imbalance problem, we present STRATA-B which aims to obtain blocks with (almost) equal functions for each subset. The intervals for a single dimension are constructed as follows: we start with an empty interval and extend it until it contains no more than $|\mathcal{I}|/K$ functions. Then the next interval is defined in the same fashion. After covering the current dimension, the process is repeated for the next one. Since all dimensions are not considered at once, we may not achieve a complete balance for the number of functions in blocks. However, the experiments show that this approach results in a better performance than STRATA. For example, in Figure 3.4(a), we obtain blocks with 6624, 3492, 2449, and 13585 functions, on the other hand in Figure 3.4(b), these values are 6395, 6232, 6204 and 6183 for STRATA-B.

Algorithm 3: HAMSI with L-BFGS Updates

input: x, `schedule`, η, γ and $\boldsymbol{\alpha}_{k,b}$ for all $k = 1, \ldots, K$; $b = 1, \ldots, B_k$.

$t = 1$, $S = \mathbf{0}$, $Y = \mathbf{0}$

$\mathcal{K} = [K, 1, 2, \ldots, K-1, K]$

repeat

  $\beta_t = (\eta t)^\gamma$

  $\mathcal{K} \leftarrow$ SetSchedule($\mathcal{K}$, `schedule`)

  for $\ell = 1, 2, \ldots, K+1$ **do**

    $k = \mathcal{K}[\ell]$

    for $b = 1, 2, \ldots, B_k$ **do in parallel**

      $g_{\boldsymbol{\alpha}_{k,b}} = \nabla f_{k,b}(x_{\boldsymbol{\alpha}_{k,b}})$

      if $\ell = 1$ **then**

        $\bar{g}_{\boldsymbol{\alpha}_{k,b}} = g_{\boldsymbol{\alpha}_{k,b}}$, $\bar{x}_{\boldsymbol{\alpha}_{k,b}} = x_{\boldsymbol{\alpha}_{k,b}}$

      $x_{\boldsymbol{\alpha}_{k,b}} = x_{\boldsymbol{\alpha}_{k,b}} - \frac{1}{\beta_t}(\sigma g_{\boldsymbol{\alpha}_{k,b}} + W_{\boldsymbol{\alpha}_{k,b}} N W_{\boldsymbol{\alpha}_{k,b}}^\top g_{\boldsymbol{\alpha}_{k,b}})$

  $s = x - \bar{x}$, $y = g - \bar{g}$,

  if $s^\top y > 0$ **then**

    $\sigma = \frac{s^\top y}{y^\top y}$

    $S = [S(:, 2:M), s]$, $Y = [Y(:, 2:M), y]$, $W = [S,\ \sigma Y]$

    $C = Y^\top Y$, $R = \mathrm{triu}(S^\top Y)$, $D = \mathrm{diag}(R) I$

    $N = \begin{bmatrix} R^{-\top}(D + \sigma C) R^{-1} & -R^{-\top} \\ -R^{-1} & 0 \end{bmatrix}$

  $t \leftarrow t + 1$

until *convergence or* $t >$ *max_ epochs*

3.4 Example Implementation

In this part, we give a particular implementation of HAMSI where BFGS quasi-Newton update rule is used to approximate Hessian matrices. The algorithm for this version is given in Algorithm 3. Specifically, we use limited memory BFGS (L-BFGS) in inner iterations to construct and solve the quadratic models [15]. The advantage of L-BFGS is that it does not require to form any $|\mathcal{J}| \times |\mathcal{J}|$ matrices and $\mathcal{O}(|\mathcal{J}|^2)$ operations. Instead, it allows to compute $(H^{(t)} + \beta_t I)^{-1} v$ for a given vector v and the required memory becomes $\mathcal{O}(M|\mathcal{J}|)$, where M is the memory size. In the general version of HAMSI (Algorithm 1), this part corresponds to the exact solution of the subproblem.

To compute approximate Hessian by using Quasi-Newton updates, in addition to the difference of two consecutive iterates, the difference of the gradients should be evaluated. In the given algorithm, Algorithm 3, line 12 corresponds to keeping the previous iterates and gradients. Then, in 14, the difference of iterates s, the difference of gradients y are computed. L-BFGS algorithm updates the approximate Hessian by using M of these difference vectors s and y. The resulting matrices sized $|\mathcal{J}| \times M$ are denoted by S and Y in the algorithm. The related part and computing approximate Hessian is given from line 14 to line 19. For details about this part, we refer to [15].

The selection of subsets is done with the SetSchedule function. This function takes all the subsets in the same cyclic order which correspons to our deterministic version. The selection of subsets randomly without replacement does also exist in the literature (*cf.* [34]). For our numerical experiments given in Section 4.4, the SetSchedule function is used in two ways to obtain *deterministic* (`det`) and *stochastic* (`stoc`) results. The deterministic order is the same as in Algorithm 1, it uses the one cyclic left-shift. The illustration of SetSchedule function for the stochastic version (shuffling) is given below for $K = 5$:

Iteration	Deterministic	Stochastic
t	$\mathcal{K} = [3, 4, 5, 1, 2, 3]$	$\mathcal{K} = [3, 1, 4, 5, 2, 3]$
$t+1$	$\mathcal{K} = [4, 5, 1, 2, 3, 4]$	$\mathcal{K} = [2, 4, 5, 3, 1, 2]$
$t+2$	$\mathcal{K} = [5, 1, 2, 3, 4, 1]$	$\mathcal{K} = [3, 5, 2, 1, 4, 3]$
$\vdots$	$\vdots$	$\vdots$

We parallelize all the vector and matrix operations in HAMSI. Additionally, for the parallel computation of the gradient vector, we employ the methods given in

Section 3.3. To make the algorithm more practical, we apply a lazy update $x = x - \frac{1}{\beta_t}(\sigma g + WNW^\top g)$ once for every iteration at line 7, instead of the updates in line 13. Moreover, we do this update in parallel from right-to-left to improve the cost and memory efficiency of HAMSI. The right-to-left order corresponds to the order in $W(N(W^\top g))$. With this approach, we do not need to store $WNW^\top$ and instead of expensive matrix-matrix multiplications, we only perform three matrix-vector multiplications.

These operations are parallelized by block-partitioning of the columns and the rows of $W^\top$ and W, respectively. Additionally, to avoid expensive shift operations for S and Y (at line 17), at every outer iteration, we only update the corresponding row/column of $R = \text{triu}(S^\top Y)$ which the new s and y vectors are inserted on top of an existing column.

3.5 Computational Study

Dataset	Algorithm	`schedule`	Average Final RMSE Value				
			Hogwild	Color	Color-B	Strata	Strata-B
$1M$ ratings – 6040 users – 3883 movies (25 seconds)	mb-GD	`det`	3.1074	3.1061	3.0845	2.5315	2.4588
		`stoc`	3.1433	3.1470	3.1003	2.5325	2.4650
	HAMSI	`det`	0.6901	0.6955	0.7102	0.6133	0.6022
		`stoc`	0.6900	0.7987	0.8017	0.6088	0.5994
$10M$ ratings – 71567 users – 10681 movies (250 seconds)	mb-GD	`det`	4.3167	4.2676	4.2617	4.0029	3.4088
		`stoc`	4.3009	4.2863	4.2801	4.0035	3.4094
	HAMSI	`det`	0.9279	1.0181	0.8941	0.8923	0.8643
		`stoc`	0.9207	1.1357	1.1229	0.8988	0.8652
$20M$ ratings – 138493 users – 26744 movies (500 seconds)	mb-GD	`det`	4.8655	4.8051	4.8000	4.8093	3.8890
		`stoc`	4.8641	4.8279	4.8142	4.8091	3.8975
	HAMSI	`det`	1.0170	1.1117	0.9521	1.0113	0.9042
		`stoc`	1.0112	1.2944	1.2220	1.0231	0.9035

Table 3.1: Root-mean square errors for mb-GD and HAMSI (deterministic and stochastic version) with different parallelization techniques.

We use the MovieLens datasets with different ratings (1M, 10M, 20M) for the ex-

periments [39]. The number of users and movies for each dataset is given in the first column of Table 3.1. Here, we aim to factorize User $\times$ Movie matrices such that the inner dimensions of the factor matrices is 50. We compare the performance of our algorithm, HAMSI, with mini-batch gradient descent (mb-GD) algorithm. The implementation of these two methods are similar to each other with some minor differences. First, mb-GD requires a different schedule order $\mathcal{K}$ which is sized K and order is permutation of $\{1, 2, \ldots, K\}$. Second difference is the update rule, instead of (line 13), mb-GD uses the stochastic gradient update equation $x \leftarrow x - \frac{1}{\beta_t} g$. Meanwhile, for the first iterations of HAMSI, the full S and Y matrices are not available yet and the gradient descent update is employed at these iterations. Since mb-GD does not use approximate Hessian, the lines 14–19 are ignored for the related experiments. There also exists some constant parameters in the algorithms, we use them as $(\eta = 0.001, \gamma = 0.51)$ for mb-GD and $(\eta = 0.06, \gamma = 0.51)$ for HAMSI.

The experiments are grouped into two sets which respect to the stopping rule. The first set of experiments, (I), considers fixed wallclock time, and the second one, (II), considers fixed number of iterations. For (I), the algorithms run for 25, 250 and 500 seconds for the datasets 1M, 10M and 20M, respectively. At the end of each experiment, we report the root-mean-square error (RMSE) to define the training error. All parallelization schemes mentioned in Section 3.3 with $1, 2, 4, 8$ and 16 threads are used and each experiment is repeated three times. At each repetition, we generate the initial solutions by employing random seeds. Thus, we perform 900 experiments in total (3 (datasets) $\times$ 2 (algorithms) $\times$ 5 (parallelization scheme) $\times$ 5 (#threads) $\times$ 3 (random seeds) $\times$ 2 (scheduling)). Additionally, the number of subsets K is defined as 20 for Hogwild, Color and Color-B. For Strata and Strata-B, it is determined as the number of threads automatically.

The results for the RMSE averages for the first experiment set (I) is given in Table 3.1. The parallel processing is employed with 16 threads for this experiments. Here, since we perform the experiments for a fixed time, the performance of algorithms is almost defined by the parallelization technique. We observe that Strata-B gives the best results in the deterministic setting. Moreover, Color-B results in better RMSE than Color in both deterministic and stochastic version. When we compare the performances of HAMSI and mb-GD, it is clearly seen that HAMSI outperforms

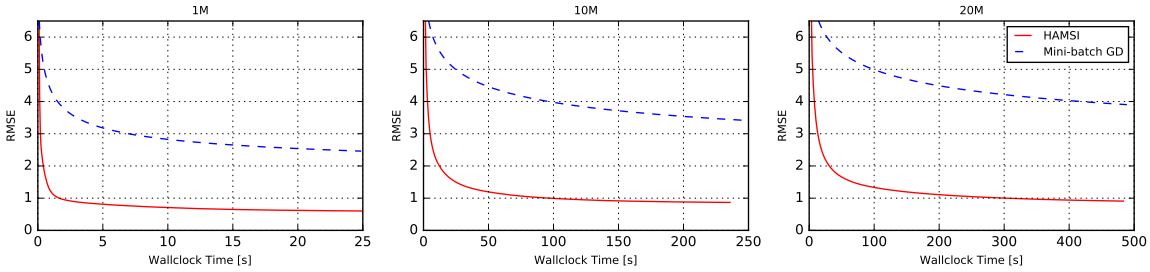


Figure 3.5: Convergence of mb-GD and HAMSI in terms RMSE values with 16 threads.

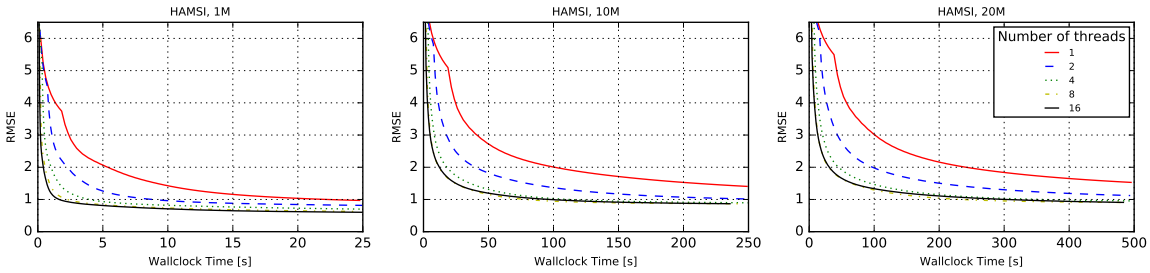


Figure 3.6: Convergence behaviors of HAMSI when the number of threads is increased.

mb-GD although mb-GD can perform more iterations because of only using the first order information.

For further comparison of mb-GD and HAMSI, we compare the change in RMSE values with respect to wallclock time. The related plots are given in Figure 3.5. Here, we pick STRATA-B for parallelization since it the best performing one in Table 3.1 in the deterministic setting. We observe that HAMSI achieves faster improvement in addition to the better RMSE value at final step. By looking at these results, we can also conclude that HAMSI obtains an important performance gain with the help of the second order information.

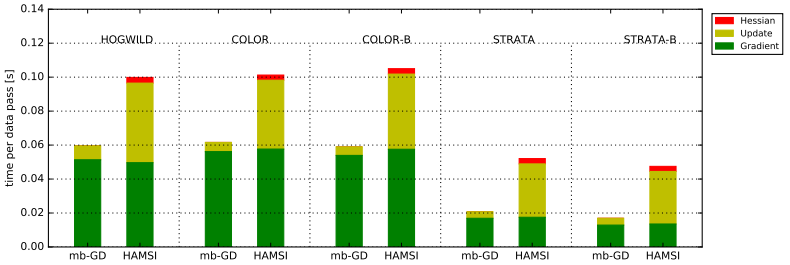
To discuss the effects of number of threads over the convergence of HAMSI, we repeat the experiments with various number of threads. For STRATA-B, we keep the number of threads and the batch size equal to each other. Thus, increasing the number of threads results in smaller batch size, i.e, higher variance of the gradient estimation. However, the results of experiments given in Figure 3.6 show that this variance does not have a significant effect on the performance of HAMSI. Conversely, the performance becomes better with the increasing number of threads.

In the next set of experiments, we analyze the efficiency of the parallelization meth-

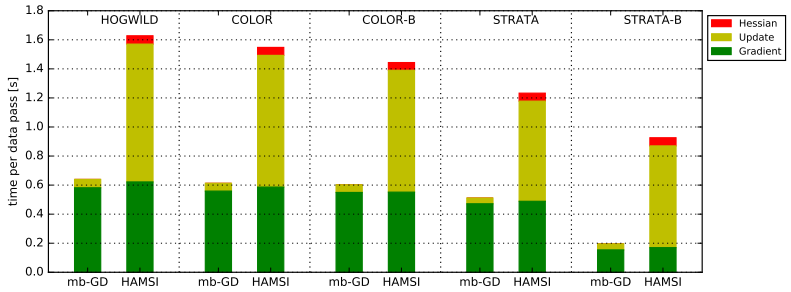
ods. Here, we use a partitioning for the cost of an outer iteration to observe the effect of parallelization schemes clearly. The partitions for Algorithm 3 are as follows:

- data pass and gradient computation (line 10),
- performing the updates (line 13),
- maintaining the approximate Hessian (lines 14–19).

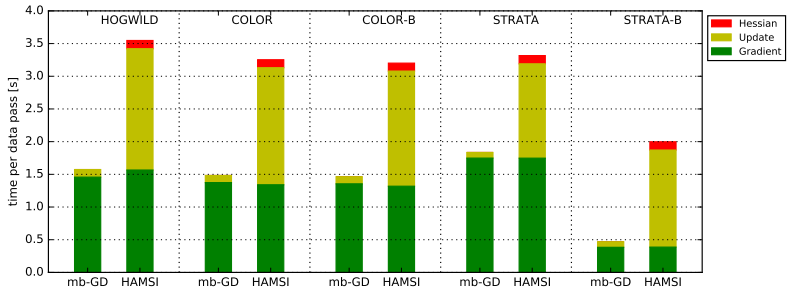
In Figure 3.7, we show the time spent in each of these three phases for different parallelization schemes with 16 threads. As it can be seen clearly, the Hessian approximation is much cheaper than the gradient computation and update parts. Also, as expected, the main time consuming phase for mb-GD is gradient computation. Thus, by using parallelization in gradient-based optimization, it is possible to increase the effectiveness of the algorithms. Although the iterations for mb-GD is cheaper and the algorithm can perform more iterations than HAMSI during the same time interval, the final RMSE values of HAMSI is much better than the values of mb-GD as it can be seen from Table 3.1 and Figure 3.5.



(a) 1M



(b) 10M



(c) 20M

Figure 3.7: Hessian computation, update, and gradient computation time for an outer iteration of mb-GD and HAMSI with 16 threads.

Chapter 4

Differentially Private Gradient-Based Algorithms

Differential privacy roughly promises securing an individual's data while still revealing useful information about a population [23]. In other words, the data analyst guarantees that the data owner will not be effected while using the data of any individual for analysis. The privacy is not a concern when the data owner and the data analyst are the same. However, in real world, it happens that a single analysis is constructed over the data gathered from different data owners. Then, privacy becomes an issue. Here, we take this scenario into consideration and try to find a solution for the problem over the data collected from multiple data contributers by an analyst.

The first part of this chapter is reserved for privacy of two first order deterministic accelerated algorithms which use full data to compute the gradient. We propose a variant of the gradient descent method as our first algorithm. For the ease of explanation and clarity, we will sacrifice generality in the text and put the optimization problem in a certain context. The outline of our first algorithm is as follows: At every iteration of the algorithm, the data holder computes a gradient vector by using the full data and releases the noisy version of the gradient. This released vector is then used to obtain a smoothed overall gradient which is finally used to update the parameter estimate. Once the noisy gradient is released; it does not matter in terms of data privacy whether the data holder or the analyst performs the update with the smoothed gradient. Since the analyst cannot know the gradient without noise, the privacy is protected with the help of random noise. As an illustration of this idea, we solve the regularized logistic

regression problem.

The second deterministic private algorithm is based on the the multistage accelerated stochastic gradient algorithm (M-ASG) which is presented in [3]. In their setting, the full gradient is used with noise which corresponds to stochastic gradient. In other words, although the algorithm is called "stochastic", there is not a randomness about the gradient computation and the complete data is used at each iteration. Similarly, we present the differentially private version of M-ASG in the deterministic setting. However, we also consider stochastic M-ASG which is constructed by sampling the data. To avoid confussion, we denote the original deterministic algorithm with noisy gradient by M-ASG and the stochastic version by stocMASG. In M-ASG, they divide the number of iterations into stages and at each stage, accelerated gradient method is run with stage related parameters and noisy gradient. Although their analysis takes the noisy gradient into consideration, the noise variance and distribution is not constructed with respect to differential privacy. There, the aim of noise is to consider the case without exact gradient information. Our purpose is to introduce a differentially private multistage accelerated algorithm by using not only stage but also noise related parameters. We also follow a multistage approach at noise variance calculation and present a stage related noise division scheme.

The second part of this chapter is reserved for the stochastic versions of the algorithms mentioned above. While dealing with differential privacy, it is important to find a way to decrease the amount of random noise or improve the performance of the algorithm. One of the most popular and easy way to achieve this aim is subsampling. It is known that the uncertainity coming from the random selection of data helps to improve the differential privacy. In this part, instead of using the full data, we only use a randomly chosen sample. We aim to decrease the variance of noise with the help of privacy amplification theorems for subsampling. In addition to the random sampling, we also try the online version of DP-GDwS. This version is based on using the data as distict subsets which are taken in a deterministic setting. Although the privacy amplification effect of sampling does not exist for this concept, using disjoint samples from the data helps to decrease the noise variance.

There exists studies in the literature regarding the use of stochastic gradient descent (SGD) in a differentially private way, (cf, [17, 76]). They study the setting in which

the final output of an optimizer is revealed after it is perturbed with noise. This perturbation ensures differential privacy in [17]. Unlike [17], we are interested in a setting where the intermediate updates of our algorithm are also available to the analyst and differential privacy is ensured by perturbing the gradient vectors rather than the final output. The latter setting may be more suitable since gradient-based algorithms are well known to have convergence guarantees with noisy gradients (provided that they are unbiased). Also, the setting in which the intermediate steps are visible can be extended for scenarios where more than one data holder collaborate to perform a gradient-based algorithm on the union of their data sets.

The work in [76] is more relevant to ours in the sense that the authors are interested in the differential privacy of their stochastic gradient descent algorithm where the intermediate steps are revealed. They propose using stochastic gradient descent with noisy gradients calculated from disjoint mini-batches. This procedure does protect the privacy of the data; however, because of the high variance, it can restrict the number of iterations that is a crucial component for an optimization algorithm.

In sum, this chapter aims to present private and still convergent algorithms by decreasing the required noise. To achieve this, we use some existing methods that are used to accelerate GD and SGD. Our main adjustments and related contributions are as follows:

- First, we present our novel algorithm DP-GDwS, a smoothing method that uses the previous gradients besides the current gradient to take a step. With the help of smoothing approach, we achieve to decrease the noise variance and our algorithm satisfy a given privacy level with less noise.

  Similar ideas are used to speed-up the convergence of gradient descent before. For instance, momentum-based methods [80], and accelerated gradient methods [59] are in this context. However, to the best of our knowledge, using smoothing in optimization with differential privacy similar to our setting has not been tried before.

- We also present a differentially private multistage accelerated gradient method (DP-MAG), which is based on M-ASG [3]. For this algorithm, we divide the privacy level into stages and compute the related noise variance specific to each

stage. This means that variance of the noise differs from iteration to iteration. This is an extension.

- DP-GDwS algorithm can be classified as a variant of heavy ball method with some special adjustments. By using this fact, we analyze our method with the help of the linear dynamical system approach. For the convergence analysis of DP-MAG, we present a theorem by following the similar steps as in the analysis of M-ASG [3].

- In the second part, we propose the stochastic versions of DP-GDwS and DP-MAG by employing sampling to improve the performance. In addition to mini-batch approach, we also present online version of DP-GDwS where we take disjoint buckets from the data. Although we lose the advantages of subsampling with this approach, we obtain an efficient algorithm with the help of disjoint selection of buckets.

- Last, we provide stepsize formulas for our algorithms. It is important to take the effect of the random noise into consideration in a principled way while finding the direction. We use this idea to improve the numerical performances of both deterministic algorithms DP-GDwS and DP-MAG, and their stochastic versions DP-SGDwS and DP-SMAG. The simulation studies show that we do achieve better performance than the heavy ball and accelerated methods with classical stepsize formulas.

4.1 Preliminaries

Let I_d and 0_d denote $d \times d$ identity and zero matrices, respectively. The Kronecker delta function $\delta[k]$ is defined as follows: $\delta[0] = 1$ and $\delta[k] = 0$ for any integer $k \geq 1$. For matrix $A \in \mathbb{R}^{d \times d}$, trace of A is shown as $Tr(A)$ and the transpose of A is A^T. The spectral radius of A is the largest absolute value of its eigenvalues and denoted by $\rho(A)$. Let $\mathcal{S}^m$ denote all $m \times m$ symmetric matrices and similarly let $\mathcal{S}^m_+$ be the set of all $m \times m$ symmetric and positive definite matrices. The Kronecker product of two matrices A and B is shown as $A \otimes B$.

Definition 4.1.1. *A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is L_f-smooth on $S \subset \mathbb{R}^d$ if for all $x, y \in S$, the following inequality holds*

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L_f \|x - y\|_2.$$

Definition 4.1.2. *For scalars $0 < \mu < L$, we define $S_{\mu,L}(\mathbb{R}^d)$ as the set of continuously differentiable functions $f : \mathbb{R}^d \to \mathbb{R}$ which are strongly convex with modulus μ and Lipschitz continuous with L, satisfying*

$$\frac{L}{2}\|x - y\|^2 \geq f(x) - f(y) - \nabla f(y)^T(x - y) \geq \frac{\mu}{2}\|x - y\|^2.$$

The ratio $\kappa = L/\mu$ is called the condition number. The following lemma is well-known; see [60] (Theorem 2.1.12) for the proof.

Lemma 4.1.1. *If $f \in S_{\mu,L}(\mathbb{R}^d)$, then for every $x, y \in \mathbb{R}^d$, we have*

$$(\nabla f(x) - \nabla f(y))^T(x - y) \geq \frac{\mu L}{\mu + L}\|x - y\|^2 + \frac{1}{\mu + L}\|\nabla f(x) - \nabla f(y)\|^2.$$

4.1.1 Gradient-based Optimization

As mentioned in the first chapter, we are dealing with the unconstrained, strongly convex optimization problem of the form

$$\min_x \sum_{i \in \mathcal{I}} f_i(x), \tag{4.1}$$

where x is a parameter vector and f_i are a collection of functions and each i corresponds to a single element from $\mathcal{I}$.

For $x \in \mathbb{R}^d$, we define the gradient vector for $\mathcal{I} = \{1, 2, ..., n\}$

$$\nabla f(x) = \sum_{i=1}^{n} \nabla f_i(x). \tag{4.2}$$

Using this notation, we can further define the subsample gradient for some subset of data $B \subseteq \{1, \ldots, n\}$ with $|B| = m$

$$\nabla f_B(x) := \sum_{i \in B} \nabla f_i(x). \tag{4.3}$$

In order to solve the problem (4.1) with the basic gradient method, we use the following update at iteration t,

$$x_{t+1} = x_t - \alpha_t \nabla f(x_t), \tag{4.4}$$

where α_t is the learning rate.

As α_{HB} and β_{HB} are the learning rate and momentum parameter respectively, the update rule for HB [66] at iteration t is

$$x_{t+1} = x_t - \alpha_{HB}\nabla f(x_t) + \beta_{HB}(x_t - x_{t-1}). \tag{4.5}$$

For NAG [59], the iterations can be written as

$$\begin{aligned} x_{t+1} &= z_t - \alpha_{AG}\nabla f(z_t), \\ z_t &= (1+\beta_{AG})x_t - \beta_{AG}x_{t-1}, \end{aligned} \tag{4.6}$$

where α_{AG} and β_{AG} are the learning rate and the momentum parameter, respectively.

Similarly, in order to solve problem (4.1) with a mini-batch stochastic gradient method, we use the following update at iteration t with batch B_t,

$$x_{t+1} = x_t - \alpha_t \nabla f_{B_t}(x_t), \tag{4.7}$$

where α_t is the learning rate. Note that when B_t is a singleton, then (4.7) corresponds to the standard stochastic gradient method. We can define stochastic versions of the accelerated gradient and the heavy ball methods in the same fashion, i.e., replacing the gradients with their minibatch estimates.

4.1.2 Differential Privacy

We are interested in modifying the steps of first order optimization algorithms to have privacy-preserving updates. Our setting is as follows: The data holder makes public the iterates $\{x_t\}_{0\leq t\leq T}$ for a total of T iterations. Also, α_t is known. If the data holder applies the related update of the method directly, the vectors $\nabla f(x_t)$ are revealed. This violates privacy since the revealed terms are deterministic functions of the data. Therefore, due to privacy concerns, some random version of the gradient has to be used. Differential privacy quantifies the privacy level that one guarantees by such randomizations.

The setting of differential privacy considers those mechanisms as randomized algorithms. A randomized algorithm takes an input dataset $D \in \mathcal{D}$ and returns the random output $A_D \in \mathcal{X}$. Such an algorithm can be associated to a function $\mathcal{A} : \mathcal{D} \to \mathcal{X}$ that maps a data set from $\mathcal{D}$ to a probability distribution $\mathcal{A}(D) \in \mathcal{X}$ such that the random output $A_D \sim \mathcal{A}(D)$.

Differential privacy basically requires that the output of an algorithm does not change probabilistically when one record of the data is changed. For datasets D_1 and D_2, let $h(D_1, D_2)$ be the Hamming distance between D_1 and D_2, that is the number of different elements between the datasets. A differentially private algorithm ensures that $\mathcal{A}(D_1)$ and $\mathcal{A}(D_2)$ are 'not much different' if $h(D_1, D_2) = 1$. This 'difference' is expressed mathematically by the following formal definition of differential privacy.

Definition 4.1.3 (ϵ- Differential Privacy)**.** (Definition 1, [24]) *A randomized algorithm $\mathcal{A}$ with set of input data sets $\mathcal{D}$ and range for its output $\mathcal{X}$ is ϵ- differential private if for all data sets $D_1, D_2 \in \mathcal{D}$ differing on at most one element i.e. $h(D_1, D_2) \leq 1$, and all $X \subseteq \mathcal{X}$,*

$$\mathbb{P}[A_{D_1} \in X] \leq e^{\epsilon}\mathbb{P}[A_{D_2} \in X]. \tag{4.8}$$

Most existing differentially private methods perturb certain functions of data with suitably chosen random noise. The amount of this noise is related to the maximum amount of change in the functions of the data when one single entity of the data is changed. This leads to the definition of sensitivity.

Definition 4.1.4 (Sensitivity)**.** (Definition 3.1, [27]) *For a function on datasets $\varphi : \mathcal{D} \mapsto \mathbb{R}^k$, $k \geq 1$, the L_1-sensitivity of φ is defined as*

$$S_1^{(\varphi)} = \max_{D_1, D_2 \in \mathcal{D}: h(D_1, D_2)=1} ||\varphi(D_1) - \varphi(D_2)||_1 \tag{4.9}$$

for all D_1, D_2 differing at most one element.

One of the most widely used differentially private mechanisms is the Laplace mechanism, which we will use in our proposed method also.

Theorem 4.1.2 (Laplace mechanism)**.** (Theorem 1, [24]) *Given function $\varphi : \mathcal{D} \mapsto \mathbb{R}^k$, the mechanism $\mathcal{A}_\varphi$, which adds independently generated noise with Laplace distribution $\text{Lap}(S_1^{(\varphi)}/\epsilon)$ to each of the k output terms, enjoys ϵ-differential privacy.*

Our methods, which we present in the following sections, use the Laplace mechanism at every iteration. As in our method, with every repetition of the mechanism on the same data, new bits of information are given to the adversary, which causes further privacy loss. The theorem below gives a formal explanation.

Theorem 4.1.3 (Composition)**.** (Corollary 3.15, [27]) *Let $\mathcal{A}_i$ each provide ϵ_i-differentially privacy. The sequence $(\mathcal{A}_1, \ldots, \mathcal{A}_T)$, whose output is the concatenation of the outputs of the individual algorithms, provides $\sum_{i=1}^{T} \epsilon_i$- differentially privacy.*

Next, we cite a result from the literature regarding our method. Improvement in differential privacy by means of random subsampling is quantified in the following theorem. This theorem is based on the proof of Lemma 34 given in [86]. We obtain a tighter bound and the proof is given in Appendix A.1.

Theorem 4.1.4 (Subsampling)**.** *Let $\mathcal{M} : \mathcal{Y}^m \to \mathcal{X}$ be a ϵ-differentially private algorithm. Then, the algorithm $\mathcal{M}' : \mathcal{Y}^n \to \mathcal{X}$ that first selects a random subsample of m distinct rows from its input data $y_{1:n} \in \mathcal{Y}^n$ and then runs $\mathcal{M}$ on the subsample is ϵ'-differentially private where*

$$\epsilon' = 2(e^{\epsilon} - 1)\frac{m}{n}. \tag{4.10}$$

Although we improve the theorem given in [86] by obtaining a tighter bound for privacy under sampling, there is another bound which is even tighter than ours for $\epsilon \leq 1$. Thus, the main theorem that we use for our numerical study (when $\epsilon \leq 1$) is given below.

Theorem 4.1.5. (Lemma 2.2, [4]) *Over a domain of data sets $\mathcal{Y}^n$, if an algorithm $\mathcal{M}$ is $\epsilon' \leq 1$ differentially private, then for any data sets $Y \in \mathcal{Y}^n$, executing $\mathcal{M}$ on uniformly random γn entries of Y ensures $2\gamma\epsilon'$-differential privacy.*

4.1.3 Dynamical System Approach

Reformulation of algorithms as linear dynamical systems is a popular approach to analyze the first order methods [30,40,48]. The standard convergence analyses are hard to interpret, and they are different for each algorithm. These methods also require the exact gradient information, which is not available for stochastic algorithms or approaches emloying noisy gradients. Dynamical system approach aims to present a systematical way to understand the relationship between the algorithm parameters, the convergence rate, and the gradient noise (if it exists).

A discrete-time linear dynamical system can be expressed as

$$\begin{aligned} \xi_{t+1} &= A\xi_t + Bu_t, \\ y_t &= C\xi_t + Du_t, \\ u_t &= \phi(y_t), \end{aligned} \tag{4.11}$$

where A, B, C and D are system matrices, ξ_t is the state, y_t is the output and u_t is the input [48]. The function ϕ is the feedback rule and for our case it corresponds to the nonlinear gradient since we are dealing with strongly convex objective.

Example 4.1.1. *For GD method, the iterations are in the following form*

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

so the system matrices are as follows:

$$\begin{aligned} A &= I_d, \\ B &= -\alpha I_d, \\ C &= I_d, \\ D &= 0_d \end{aligned} \tag{4.12}$$

And the state vector in dynamical system representation (4.11) *is* $\xi_t = x_t$ *for GD method.*

For strongly convex quadratic objective functions, the analysis is easier since the gradient and the resulting system is linear. In this case, the (asymptotic) convergence rate can be obtained by computing the spectral radius of the state-transition matrix [48].

To illustrate the idea, we provide the convergence rate computation for strongly convex quadratic objectives in the following example.

Example 4.1.2. ([48], Page 5) *Assume that the objective function* f *is strongly convex quadratic function* $f(y) = \frac{1}{2}y^T Q y - p^T y + r$*, where* $mI_d \leq Q \leq LI_d$ *and the system matrices are* A, B, C *and* D *where* $D = 0$*. When we run a first order optimization algorithm over the problem, the system becomes*

$$\xi_{t+1} = A\xi_t + Bu_t,$$

$$y_t = C\xi_t,$$

$$u_t = \nabla f(y_t) = Q(y_t - y^*).$$

To talk about the optimality of y^*, $u^* = \nabla f(y^*) = 0$ *and this requires* $y^* = C\xi^*$ *and* $\xi^* = A\xi^*$ *where* $\xi^* = x^*$. *Thus, we obtain*

$$\xi_{t+1} - \xi^* = (A + BQC)(\xi_t - \xi^*).$$

A necessary and sufficient condition to show the convergence of ξ_t *to* ξ^* *is that the spectral radius of* $T = (A+BQC)$ *is strictly less than 1. If we denote the spectral radius of* T *as* $\rho(T)$, *for any* $\epsilon > 0$ *and all sufficiently large* t, $\rho(T)^t \leq \|T^t\| \leq (\rho(T) + \epsilon)^t$. *Then, the convergence rate of the algorithm can be bounded as*

$$\|\xi_t - \xi^*\| = \|T^t(\xi_0 - \xi^*)\| \leq \|T^t\|\|\xi_0 - \xi^*\| \leq (\rho(T) + \epsilon)^t\|\xi_0 - \xi^*\|$$

which shows the relationship between convergence rate and the spectral radius.

On the other hand, for other type of problems with strongly convex objectives, a linear dynamic system with nonlinear feedback approach (due to the nonlinear gradient) can be used as in [2]. In [2], the authors derive upper bounds on the robustness with the help of Lyapunov functions and analyze GD and NAG methods.

Lyapunov theory is based on minimum energy and used to explain the convergence behaviors of algorithms. Lyapunov function is a nonnegative function representing the state of the algorithm. Its value decreases along all admissible trajectories. After constructing this function, the convergence rate of the method can be explained by relating to the decrease of the internal energy.

4.2 Momentum-Based Algorithms Using Full Gradient Descent

We first present our deterministic algorithms which use the full data to compute the gradient vector. Differential privacy of the algorithms which use full data is studied before in the literature [4, 93]. However, using full data causes higher variance of required noise for DP. To overcome this problem, we present an approach to improve the performance of the full gradient algorithms with noise related parameter selection. We apply this idea to our algorithms.

4.2.1 Gradient Descent Algorithm with Smoothing

When privacy is of concern for the optimization problem given in Section 4.1.1, one approach is to update the parameter x_t of iteration t using a noisy gradient vector

$$\widetilde{\nabla f}(x_t) = \nabla f(x_t) + \eta_t, \tag{4.13}$$

where the noise term is a Laplace distributed random variable

$$\eta_{t,i} \overset{\text{i.i.d.}}{\sim} \text{Lap}(\sigma_t), \quad i = 1, \ldots, d,$$

independent from the other noise terms and $\eta_t = [\eta_{t,1}, ..., \eta_{t,d}]$.

Although the privacy of an algorithm can be guaranteed in this way, the performance will be affected because of the noise added at each iteration. As it is mentioned before, we aim to improve the performance of the algorithms by adding less noise to satisfy the same privacy level. With this aim, we propose DP-GDwS.

Algorithm DP-GDwS

We introduce a smoothing effect to improve the differential privacy of the algorithm. More concretely, the update rule for our algorithm is

$$x_{t+1} = x_t - \alpha_t \widetilde{\nabla f}^{(t)}(x_{1:t}), \tag{4.14}$$

where the smoothed gradient estimate $\widetilde{\nabla f}^{(t)}(x_{1:t})$ is a geometrically weighted average of all the previous gradients and the current one. Specifically, letting

$$\widetilde{\nabla f}^{(0)}(x_0) = \beta \nabla f(x_0) + \eta_0,$$

we define the rest recursively as

$$\widetilde{\nabla f}^{(t)}(x_{1:t}) = (1-\beta)\widetilde{\nabla f}^{(t-1)}(x_{1:t-1}) + \beta \nabla f(x_t) + \eta_t. \tag{4.15}$$

The idea of using a smoothed gradient is to improve the privacy level with the *same amount of noise* by means of giving a smaller weight to the new gradient calculation. Note that the new information that an adversary can learn from the x_t given the previous samples $x_1, \ldots, x_{t-1}$ is

$$\beta \nabla f(x_t) + \eta_t, \tag{4.16}$$

which is a noise-added calculation. This is in contrast with $\nabla f(x_t) + \eta_t$. In the latter case, the sensitivity of the calculation is $1/\beta$ times more and therefore the privacy breach is $1/\beta$ times more.

Algorithm 4 describes our gradient-descent method for solving the generic problem formulated in (4.1) when the desired privacy level is ϵ and the maximum number of iterations is T.

Algorithm 4: DP-GDwS: Differentially private smoothed gradient descent algorithm

Input: Sensitive data $y_{1:n}$, initial value x_0, number of iterations T, total privacy breach ϵ

Output: The adversar's view: x_t, $t = 1, \ldots, T$.

for $t = 0, \ldots, T-1$ **do**

  Calculate $\nabla f(x_t) = \sum_{i=1}^{n} \nabla f_i(x_t)$

  Calculate $S_1(x_t)$ and set $\sigma_t = \frac{S_1(x_t)\beta}{\epsilon/T}$.

  Sample $\eta_t \sim \text{Lap}(\sigma_t)$

  if $t = 0$ **then**

    Initialise

$$\widetilde{\nabla f}^{(0)}(x_0) = \beta \nabla f(x_0) + \eta_0.$$

  else

    Calculate

$$\widetilde{\nabla f}^{(t)}(x_{1:t}) = (1-\beta)\widetilde{\nabla f}^{(t-1)}(x_{1:t-1}) + \beta \nabla f(x_t) + \eta_t.$$

  Update $x_{t+1} = x_t - \alpha_t \widetilde{\nabla f}(x_{1:t})$.

We now proceed to establishing the differential privacy of Algorithm 4 formally. As revealed in Proposition 4.2.1, this is simply done by combining the results of Laplace mechanism and composition.

Proposition 4.2.1. *The proposed smoothing scheme in* (4.15) *leads to an ϵ-DP private algorithm when the variance of the noise at any iteration t is*

$$\sigma_t = \frac{S_1(x_t)\beta}{\epsilon/T}, \tag{4.17}$$

where $S_1(x_t)$ is the sensitivity bound, β is the smoothing weight, T is the maximum number of iterations and n is the data size.

Proof. Observing (4.2), we see that change in one data item changes only one term in $f(x_t)$. Therefore, the sensitivity of $f(x_t)$ is given by $S_1(x_t)$. Hence, with (4.17), the privacy loss of revealing (4.16) with σ_t at iteration t would be

$$\epsilon_0 = S_1(x_t)\beta/\sigma_t = \epsilon/T.$$

Finally, we apply Theorem 4.1.3 to conclude that the privacy loss after T iterations is ϵ. □

Determining the Stepsize

The theory regarding gradient algorithms suggest that the stepsize should be proportional to $1/L$, where L is the Lipschitz constant. Instead, to adjust the stepsize, we look at the variance of $\widetilde{\nabla f}(x_t)$ with respect to the noise terms until time t. We then suggest the following general form for the stepsize α_t

$$\alpha_t \approx \left[\mathrm{Var}\left(\widetilde{\nabla f}^{(t)} \underbrace{(x_t, \ldots, x_t)}_{t+1 \text{ times}} \right) \right]^{-1/2} \frac{1}{(T+1)^\gamma}, \tag{4.18}$$

where the term in square brackets is the average over the variances with respect to the components of the gradient vector, T is the number of iterations and $\gamma \in (0, 1]$ is a constant. Note that we consider the smoothed gradient vector evaluated with the same x_t for all iterations up to iteration t. The idea of checking the variance with the same x_t is based on the assumption that as the number of iterations grows x_t's are getting close to each other. For $x \in \mathcal{X}$, let

$$S_2(x) = \sup_{x,x'} ||f(x) - f(x')||_2$$

be the L_2-sensitivity of $f(x)$. We present a bound on the variances in Proposition 4.2.2; the proof is left to Appendix A.2 for the case where the gradient vector is computed over the sampled data. However, the proof is satisfied also for the deterministic case with a simple adjustment (the variance of the full gradient is taken instead of the variance of the sampled gradient in (A.5)).

Proposition 4.2.2. *For given $x_t \in \mathcal{X}$, smoothing parameter $0 < \beta \leq 1$, and $\eta_i \overset{\text{i.i.d.}}{\sim} \text{Lap}(\sigma)$ for $i = 1, \ldots, t$, the trace of the covariance of $\widetilde{\nabla f}^{(t)}(x_t, \ldots, x_t)$ with respect to noise vector $\eta_{1:t}$ is bounded as*

$$\text{Var}\left(\widetilde{\nabla f}^{(t)} \underbrace{(x_t, \ldots, x_t)}_{t+1 \text{ times}}\right) \leq \left(\frac{S_1(x_t)^2}{4} + \frac{2d\sigma^2}{\beta^2}\right)\frac{\beta}{2-\beta}. \tag{4.19}$$

In the light of the above proposition, if the privacy loss and the number of iterations are chosen as ϵ and T, respectively, we choose the stepsize is as follows:

$$\alpha_t = \left(\left(\frac{S_2(x_t)^2}{4d} + \frac{2S_1(x_t)^2}{\left(\frac{\epsilon}{T}\right)^2}\right)\frac{\beta}{2-\beta}\right)^{-1/2}\frac{0.25}{(T+1)^{1/2}}. \tag{4.20}$$

Note that we have T, the maximum number of iterations, in the stepsize formula. This makes sense in our setting, since due to privacy concerns, it is not possible to run the algorithm infinitely. Since T is fixed and we compute the sensitivity bound as independent from the iteration vector x_t, our stepsize formulation is fixed and does not change during iterations. Thus, the final form of the stepsize is

$$\alpha = \left(\left(\frac{S_2^2}{4d} + \frac{2S_1^2}{\left(\frac{\epsilon}{T}\right)^2}\right)\frac{\beta}{2-\beta}\right)^{-1/2}\frac{0.25}{(T+1)^{1/2}}. \tag{4.21}$$

Convergence Rate Analysis

The iterations of DP-GDwS can be written as

$$x_{t+1} = x_t - \alpha\beta\nabla f(x_t) + (1-\beta)(x_t - x_{t-1}),$$

which is a special case of the heavy ball method for $\alpha\beta = \alpha_{HB}$, $1-\beta = \beta_{HB}$ when the heavy ball iterates are as in (4.5).

For the dynamical system representation of HB, the system matrices can be written as follows:

$$A = \begin{bmatrix} (1+\beta_{HB})I_d & -\beta_{HB}I_d \\ I_d & 0_d \end{bmatrix},\ B = \begin{bmatrix} -\alpha_{HB}I_d \\ 0_d \end{bmatrix}, C = \begin{bmatrix} I_d & 0_d \end{bmatrix},\ D = 0_d.$$

Since we are dealing with differential privacy, instead of working with the full gradient, we will take the noisy version as $\nabla f(x_t) + \eta_t$ where $\eta_t \in \mathbb{R}^d$ is the Laplace distributed noise. Then, the algorithm can be written as:

$$\begin{aligned} \xi_{t+1} &= A\xi_t + B(u_t + \eta_t), \\ h_t &= C\xi_t, \\ u_t &= \nabla f(h_t), \end{aligned} \tag{4.22}$$

where A, B, C and D are as before. To explain the relation between ξ_t and x_t ($t \geq 0$), we use $x_t = T\xi_t$ where $T = [I_d \quad 0_d]$ (for HB). Similar to [2], we denote $f(T\xi)$ as $\bar{f}(\xi)$ to simplify the notation, and thus, $\bar{f}(\xi_t) = f(x_t)$ for all $t \geq 0$.

To satisfy ϵ-DP, η_t is Laplace distributed with mean 0 and variance $\sigma^2 = (S_1/\epsilon_{iter})^2$. That is,

$$\mathbb{E}[\eta_t] = 0, \ \mathbb{E}[\|\eta_t\|^2] = \sigma^2.$$

Since we continuously add noise, there is a possibility that $\lim_{t\to\infty} \mathbb{E}[f(x_t)]$ may not exist. So, we consider the worst-case limiting suboptimality as in [2] to determine the robustness of the algorithm to the noise,

$$\mathcal{J} = \limsup_{t\to\infty} \frac{1}{\sigma^2}\mathbb{E}[f(x_t) - f^*], \tag{4.23}$$

where f^* is the optimal objective function value.

Next, we explain the relationship between the robustness and the parameters of DP-GDwS. As mentioned before, we will use the idea in [2] to analyze our algorihtm DP-GDwS. Their analysis is based on bounding $\mathcal{J}$ with the help of Lyapunov functions. Similar to [2], we aim to bound $\mathbb{E}[f(x_t) - f^*]$ for our algorithm, DP-GDwS, by using non-negative parameters ψ_0, R which depend on algorithm parameters, initial point and the convergence rate $0 < \rho < 1$ as follows:

$$\mathbb{E}[f(x_t) - f^*] \leq \rho^{2k}\psi_0 + \sigma^2 R, \quad \forall k \geq 0.$$

Thus R is as an upper bound on $\mathcal{J}$. We can say that the convergence rate of the expected suboptimality is ρ to the interval around 0 with radius $\sigma^2\mathcal{J}$. As it is clearly seen, the case $\sigma = 0$ corresponds to the convergence rate of $f(x_t)$ to directly f^* and the part $\sigma^2 R$ corresponds to the effect of the random noise.

Specifically, our Lyapunov function is in the following form

$$V_{P,c}(\xi) = V_P(\xi) + c(\bar{f}(\xi) - f^*), \tag{4.24}$$

where c is nonnegative constant P is a positive semidefinite matrix shown as $P \succeq 0$, and $V_P(\xi) = (\xi - \xi^*)^T P(\xi - \xi^*)$. The idea of constructing a Lyapunov function is based on determining the convergence rate by considering the change in the function value.

We know that under noisy gradient, Lemma 4.1 ([3]) which explains the evolution of $V_P(\xi_{t+1})$ along t, and Corrollary 4.2 ([3]) which characterize the difference $\mathbb{E}[V_P(\xi_{t+1})]-$

$\rho^2\mathbb{E}[V_P(\xi_t)]$ is satisfied.

Before deriving the bounds, we need some assumptions:

A.1 There exists a symmetric matrix $X \in \mathcal{S}^{m+d}$ such that

$$X \succeq \Phi(A, B, P, \rho) \tag{4.25}$$

for some $P \in \mathcal{S}_+^m$ and $\rho \in (0, 1)$ where

$$\Phi(A, B, P, \rho) = \begin{bmatrix} A^T P A - \rho^2 P & A^T P B \\ B^T P A & B^T P B \end{bmatrix}$$

A.2 For some nonnegative constants Γ and c, the same parameters ρ and X as before satisfy

$$\mathbb{E}\left[\begin{bmatrix} \xi_t - \xi^* \\ \nabla f(h_t) \end{bmatrix}^T X \begin{bmatrix} \xi_t - \xi^* \\ \nabla f(h_t) \end{bmatrix}\right] \leq c(\rho^2\mathbb{E}[\bar{f}(\xi_t) - f^*] - \mathbb{E}[\bar{f}(\xi_{t+1}) - f^*] + \sigma^2\Gamma) \tag{4.26}$$

for every $t \geq 0$.

Thus, as in Aybat et. al ([2]), by using A.1 and A.2 along with Corollary 4.2, the following inequalities are satisfied for all $t \geq 0$

$$\rho^2\mathbb{E}[V_{P,c}(\xi_t)] + \sigma^2(\mathrm{Tr}(B^T P B) + c\Gamma) \geq \mathbb{E}[V_{P,c}(\xi_{t+1})],$$

and

$$\mathbb{E}[V_{P,c}(\xi_t)] \leq \rho^{2t} V_{P,c}(\xi_0) + \frac{1 - \rho^{2t}}{1 - \rho^2}\sigma^2 R_P,$$

where $R_P = \mathrm{Tr}(B^T P B) + c\Gamma$.

We next consider the steps of our DP-GDwS algorithm (with DP noise) given by

$$x_{t+1} = (2 - \beta)x_t - (1 - \beta)x_{t-1} - \alpha\beta(\nabla f(x_t) + \eta_t).$$

Lemma 4.2.3. *Let $f \in \mathcal{S}_{\mu,L}(\mathbf{R}^d)$, then for any $\rho \in (0, 1)$,*

$$\begin{bmatrix} \xi_t - \xi^* \\ \nabla f(h_t) \end{bmatrix}^T (X_1 + (1 - \rho^2)X_2) \begin{bmatrix} \xi_t - \xi^* \\ \nabla f(h_t) \end{bmatrix} \leq \rho^2(f(x_t) - f^*) - (f(x_{t+1}) - f^*)$$
$$-\frac{L\alpha^2\beta^2}{2}\|\eta_t\|^2 + L\alpha\beta\left[(1 - \beta)(x_t - x_{t-1}) - \alpha\beta\nabla f(h_t)\right]\eta_t + \alpha\beta\nabla f(h_t)\eta_t, \tag{4.27}$$

where $X_1 = \tilde{X}_1 \otimes I_d$ *and* $X_2 = \tilde{X}_2 \otimes I_d$ *with*

$$\tilde{X}_1 = \frac{1}{2}\begin{bmatrix} -L(1-\beta)^2 & L(1-\beta)^2 & -(1-L\alpha\beta)(1-\beta) \\ L(1-\beta)^2 & -L(1-\beta)^2 & (1-L\alpha\beta)(1-\beta) \\ -(1-L\alpha\beta)(1-\beta) & (1-L\alpha\beta)(1-\beta) & \alpha\beta(2-L\alpha\beta) \end{bmatrix}$$

and

$$\tilde{X}_2 = \frac{1}{2}\begin{bmatrix} \mu & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}.$$

Proof. The iterations of our algorithm can be written as follows:

$$x_{t+1} = (2-\beta)x_t - (1-\beta)x_{t-1} - \alpha\beta\nabla f(h_t),$$

$$h_t = x_t.$$

Then from Definition 4.1.2, we can write

$$f(h_t) - f(x_{t+1}) \geq \nabla f(h_t)^T(h_t - x_{t+1}) - \frac{L}{2}\|x_{t+1} - h_t\|^2.$$

Then,

$$\begin{aligned} f(h_t) - f(x_{t+1}) &\geq \nabla f(h_t)^T(h_t - (2-\beta)x_t + (1-\beta)x_{t-1} + \alpha\beta(\nabla f(h_t) + \eta_t)) \\ &\quad -\frac{L}{2}\|(2-\beta)x_t - (1-\beta)x_{t-1} - \alpha\beta(\nabla f(h_t) + \eta_t) - h_t\|^2 \end{aligned} \tag{4.28}$$

$$\begin{aligned} &= -(1-\beta)(x_t - x_{t-1})\nabla f(h_t) + \alpha\beta\|\nabla f(h_t)\|^2 + \alpha\beta\|\nabla f(h_t)\|\eta_t - \frac{L}{2}(1-\beta)^2\|x_t - x_{t-1}\|^2 \\ &\quad + L\alpha\beta(1-\beta)(x_t - x_{t-1})(\nabla f(h_t) + \eta_t) - \frac{\alpha^2\beta^2 L}{2}\|\nabla f(h_t) + \eta_t\|^2 \end{aligned} \tag{4.29}$$

$$\begin{aligned} &= \begin{bmatrix} x_t - x_{t-1} \\ \nabla f(h_t) \end{bmatrix}^T \begin{bmatrix} -\frac{L}{2}(1-\beta)^2 & \frac{L\alpha\beta(1-\beta)-(1-\beta)}{2} \\ \frac{L\alpha\beta(1-\beta)-(1-\beta)}{2} & -\alpha^2\beta^2\frac{L}{2} + \alpha\beta \end{bmatrix} \begin{bmatrix} x_t - x_{t-1} \\ \nabla f(h_t) \end{bmatrix} - \frac{L\alpha^2\beta^2}{2}\|\eta_t\|^2 \\ &\quad + L\alpha\beta\left[(1-\beta)(x_t - x_{t-1}) - \alpha\beta\nabla f(h_t)\right]\eta_t + \alpha\beta\nabla f(h_t)\eta_t \end{aligned} \tag{4.30}$$

$$\begin{aligned} &= \frac{1}{2}\begin{bmatrix} x_t - x_* \\ x_{t-1} - x_* \\ \nabla f(h_t) \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \end{bmatrix}^T \begin{bmatrix} -\frac{L}{2}\beta^2 & \frac{L\alpha\beta(1-\beta)-(1-\beta)}{2} \\ \frac{L\alpha\beta(1-\beta)-(1-\beta)}{2} & -\alpha^2\beta^2\frac{L}{2} + \alpha\beta \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t - x_* \\ x_{t-1} - x_* \\ \nabla f(h_t) \end{bmatrix} \\ &\quad - \frac{L\alpha^2\beta^2}{2}\|\eta_t\|^2 + L\alpha\beta\left[(1-\beta)(x_t - x_{t-1}) - \alpha\beta\nabla f(h_t)\right]\eta_t + \alpha\beta\nabla f(h_t)\eta_t \end{aligned} \tag{4.31}$$

Thus,

$$\frac{1}{2}\begin{bmatrix} x_t - x_* \\ x_{t-1} - x_* \\ \nabla f(h_t) \end{bmatrix}^T \begin{bmatrix} -L(1-\beta)^2 & L(1-\beta)^2 & -(1-L\alpha\beta)(1-\beta) \\ L(1-\beta)^2 & -L(1-\beta)^2 & (1-L\alpha\beta)(1-\beta) \\ -(1-L\alpha\beta)(1-\beta) & (1-L\alpha\beta)(1-\beta) & \alpha\beta(2-L\alpha\beta) \end{bmatrix} \begin{bmatrix} x_t - x_* \\ x_{t-1} - x_* \\ \nabla f(h_t) \end{bmatrix}$$
$$-\frac{L\alpha^2\beta^2}{2}\|\eta_t\|^2 + L\alpha\beta\left[(1-\beta)(x_t - x_{t-1}) - \alpha\beta\nabla f(h_t)\right]\eta_t + \alpha\beta\nabla f(h_t)\eta_t \leq f(x_t) - f(x_{t+1}), \tag{4.32}$$

which gives the first matrix X_1. Similarly,by using Definition 4.1.2

$$\begin{aligned} f(x_*) - f(h_t) &\geq \nabla f(h_t)(x_* - h_t) + \frac{\mu}{2}\|x_* - h_t\|^2 \\ &= \nabla f(h_t)(x_* - x_t) + \frac{\mu}{2}\|x_* - x_t\|^2 \end{aligned}$$

$$f(x_*) - f(x_t) \geq \frac{1}{2}\begin{bmatrix} x_t - x_* \\ x_{t-1} - x_* \\ \nabla f(h_t) \end{bmatrix}^T \begin{bmatrix} \mu & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_t - x_* \\ x_{t-1} - x_* \\ \nabla f(h_t) \end{bmatrix}, \tag{4.33}$$

and that gives us the second matrix X_2. When we take

$$X = X_1 + (1-\rho^2)X_2,$$

and by (4.32) and multiplying (4.33) by $(1-\rho^2)$ we obtain the desired result.

□

Thus, (4.26) is satisfied and A.2 is correct for DP-GDwS. The following proposition, which is constructed based on the Proposition 4.6 in [2], helps us to find (ρ, P) pairs where ρ is the convergence rate and P is a $d \times d$ positive definite matrix. Since we are dealing with a variant of HB and using smoothing, X_1, X_2 and the bound differs, but the proof follows the same steps as the proof of Proposition 4.6 in [2].

Proposition 4.2.4. *Let $f \in \mathcal{S}_{\mu,L}(\mathbb{R}^d)$, and consider the DP-GDwS iterations. Assume there exist $\rho \in (0,1)$, $P \in \mathcal{S}_+^{2d}$, and $c_0, c \geq 0$ such that*

$$c_0 X_0 + cX(\rho) \succeq \Phi(A, B, P, \rho), \tag{4.34}$$

where

$$X_0 = \begin{bmatrix} 2\mu L C^T C & -(\mu+L)C^T \\ -(\mu+L)C & 2I_d \end{bmatrix}, \quad X(\rho) = X_1 + (1-\rho^2)X_2$$

for X_1 and X_2 defined in previous lemma. Then the following bounds hold for all $t \geq 0$:

$$\mathbb{E}[V_{P,c}(\xi_t)] \leq \rho^{2t} V_{P,c}(\xi_0) + \frac{1-\rho^{2t}}{1-\rho^2}\sigma^2\alpha^2\beta^2\left(\frac{c}{2}Ld + \mathrm{Tr}(P_{11})\right), \tag{4.35}$$

where $P_{11} \in \mathcal{S}_+^d$ is the submatrix of P formed by its first d rows and d columns.

Although solving the given matrix inequality (4.34) gives us the rate, solving this inequality becomes harder with the increasing size of d. To overcome this issue, [2] proves that finding a matrix $P \in S_+^2$ satisfying the inequality (4.35) is enough to find a convergence rate. Their corollary is generalized to our method as follows:

Corollary 4.2.5. (Corollary 4.7, [2]) *Let $f \in S_{\mu,L}(\mathbb{R}^d)$, and consider the DP-GDwS iterations with parameters α and β. Assume there exist $\rho \in (0,1)$, $\tilde{P} \in S_+^2$, $c_0 \geq 0$, and $c > 0$ such that $c_0 X_0 + cX(\rho) \succeq \Phi(A, B, P, \rho)$ with X_0 defined in Proposition 4.2.4, X_1, X_2 defined in Lemma 4.2.3, and $P = \bar{P} \otimes I_d$. Then for all $t \geq 0$,*

$$\mathbb{E}[f(x_t) - f^*] \leq \rho^{2t}\psi_0 + (1-\rho^{2t})\sigma^2\mathcal{R}_{AG}(\alpha,\beta), \tag{4.36}$$

$$\mathcal{R}_{AG}(\alpha,\beta) = \begin{cases} \frac{L\alpha^2\beta^2 d}{2(1-\rho^2)}\frac{cL+2\tilde{P}_{11}}{cL+2(\tilde{P}_{11}-\tilde{P}_{12}^2/\tilde{P}_{22})}, & \tilde{P}_{22} > 0; \\ \frac{L\alpha^2\beta^2 d}{2(1-\rho^2)}, & \tilde{P}_{22} = 0, \end{cases} \tag{4.37}$$

where $\psi_0 = \frac{1}{c}V_{P,c}(\xi_0)$. As a consequence, $\mathcal{J} \leq \mathcal{R}_{AG}(\alpha,\beta)$.

As a result, the algorithm, DP-GDwS, converges to the interval $[f^*, f^* + \sigma^2\mathcal{R}_{AG}]$ with rate ρ if there exists (ρ, P) pairs such that P is positive definite and (4.35) is satisfied. There are some approaches to find suitable ρ and P, but it is difficult to find a solution by solving the related matrix inequality analytically for HB. For NAG, it is easier since the general form of the P matrix is known in terms of algorithm parameters. On the other hand, the existing studies dealing with the rate of HB generally apply a grid search method to determine suitable (ρ, P) pairs. Since DP-GDwS is a special form of HB, we apply a grid search approach similar to the one in [48].

In [48], the authors obtain a convergence rate for HB when IQC approach is used. They consider the non-noisy setting and claim that HB fails to converge for large condition number κ values when the step size is $\alpha = \frac{4}{(\sqrt{L}+\sqrt{\mu})^2}$ which is the optimal stepsize for HB for quadratic objective. In the same paper, it is also shown that the performance of HB is better for $\alpha = \frac{1}{L}$. In case the gradient is noisy, the analysis in [2]

shows that the rate value determines the convergence rate to the interval $[f^*, f^* + \sigma^2\mathcal{R}_{AG}]$ as it mentioned before. For our algorithm DP-GDwS, we obtain the result in the left plot of Figure 4.1 for various κ values. It is clearly seen that our algorithm performs better for larger κ values. The green plot in the same figure shows the rate in case the objective function is quadratic and we see that our version obtains an improvement over the rate for quadratics.

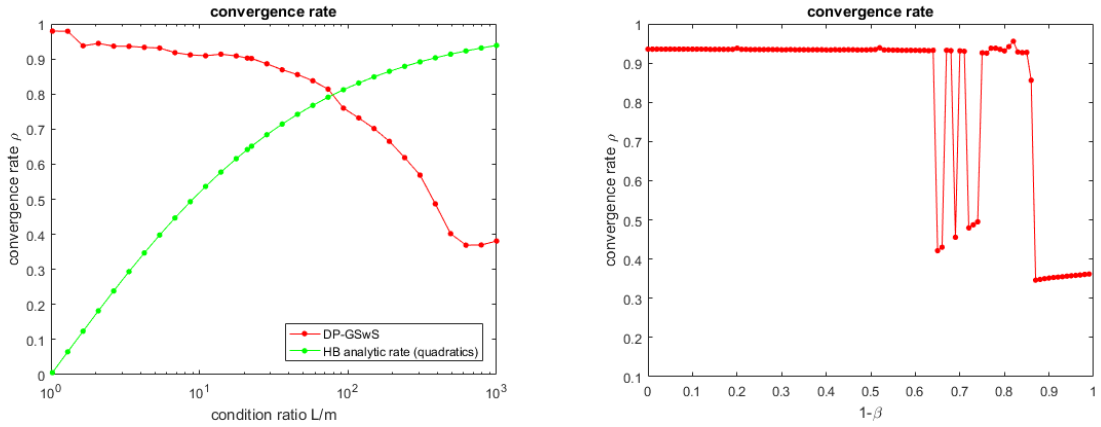


Figure 4.1: Convergence Rate for different κ values

We claim that by selecting smaller β values, the required noise is decreased and we improve the performance of the algorithm. To prove this claim, we repeat the convergence rate experiments for $\epsilon = 1$ and $\beta \in \{0.01, 0.02, ..., 0.99, 1\}$ values. The result is given in the right plot of Figure 4.1 . The x-axis represents $1 - \beta$ and y-axis is the corresponding rate value. The figure shows that increasing values of $1 - \beta$ which means the decreasing values of β result in better convergence rate until a threshold value. This supports our idea of using smoothing to improve the performance.

4.2.2 Multistage Accelerated Algorithm

In this part, we present differentially private version of M-ASG introduced in [3]. This algorithm consists of stages where accelerated gradient is used and at each iteration they compute a noisy full gradient instead of the gradient itself. The authors prove that M-ASG achieves the optimal rate both in deterministic and stochastic versions. Although their study deals with gradient with random noise, they do not consider the differential privacy context in the algorithm design. In other words, the variance of the noise is not computed by considering the sensitivity of data as in differential privacy.

Our extension considers the following:

- We add sensitivity related Laplace noise to satisfy differential privacy by taking the multistage approach into consideration. That is, instead of satisfying ϵ/T privacy for each iteration, we satisfy ϵ/K privacy at each stage where ϵ is privacy level, T is the number of iterations and K is the number of stages.

- We use a noise related stepsize at each stage by keeping the stage-related formulas as required in [3].

Algorithm DP-MAG

The original algorithm M-ASG is a multistage accelerated algorithm which uses Nesterov's accelerated gradient method with noisy full gradient [3]. The authors divide the number of iterations into stages and determine the number of iterations n_k and stepsize α_k at each stage $k \in 1, ..., K$ as

$$\begin{aligned} &n_1 \geq 1, \quad \alpha_1 = \frac{1}{L}, \\ &n_k = 2^k \lceil \sqrt{\kappa} \log(2^{p+2}) \rceil, \quad \alpha_k = \frac{1}{2^{2k} L} \end{aligned} \tag{4.38}$$

where $p \geq 1$, L is the Lipschitz constant and $\sum_{k=1}^{K} n_k = T$ with T number of iterations in total. M-ASG achieves the optimal rate and it is given in Theorem 3.4 ([3]) that the last iterate of each stage, $x_{n_k+1}^k$, satisfies the following bound for all $k \geq 1$:

$$\mathbb{E}[f(x_{n_k+1}^k)] - f^* \leq \frac{2}{2^{(p+1)(k-1)}} (\exp(-n_1/\sqrt{\kappa})(f(x_0^0) - f^*)) + \frac{\sigma^2 \sqrt{\kappa}}{L 2^{k-1}}, \tag{4.39}$$

where x_0^0 is the initial iterate.

In (4.39), σ^2 represents the noise variance which is a predefined constant, and independent from the structure of the data. This noise is not constructed with the aim of privacy, they only consider the case where an inexact gradient is available, thus the algorithm is not proven to be differentially private. On the other hand, to satisfy differential privacy a noise adding mechanism constructed by considering sensitivity bound is used in the literature as in Theorem 4.1.2. To make M-ASG differentially private, we propose some arrangements related to algorithm parameters and the noise dividing scheme to the stages.

The reason behind this noise dividing scheme is that the number of iterations at each stage increases with ratio 2^k where k is the stage number; see (4.38). On the

other hand, it is clear from the same equation that the stepsize decreases with ratio 2^{2k}. Thus, we expect a balance between the iteration number, the stepsize and the variance of noise for each stage. To keep this balance, we divide the noise into the stages equally.

Algorithm 5 describes our differentially private version of M-ASG method for solving the generic problem formulated in (1.1) when the desired privacy level is ϵ and the maximum number of iterations is T.

Algorithm 5: DP-MAG: Differentially private multistage accelerated algorithm

Input: Initial value x_0^0, the sequence of stepsizes $\{\alpha_k\}_{k=1}^K$, number of iterations at each stage $\{n_k\}_{k=1}^K$, total privacy breach ϵ

Output: The adversar's view: x_t, $t = 1, \ldots, T$.

Set $n_0 = -1$

for $k = 1; k \leq K; k = k + 1$ **do**

  Compute $\epsilon_k = \epsilon/K$

  Set $x_0^k = x_1^k = x_{n_{k-1}+1}^{k-1}$

  for $t = 1; t \leq n_k; t = t + 1$ **do**

    Calculate $S_1(x_t^k)$ and set $\sigma_{k,t} = \frac{S_1(x_t^k)}{\epsilon_k/n_k}$.

    Sample $\eta_t^k \sim \text{Lap}(\sigma_{k,t})$

    Set $\beta = \frac{1-\sqrt{\mu/L}}{1+\sqrt{\mu/L}}$

    Set $y_t^k = (1 + \beta_k)x_t^k - \beta_k x_{t-1}^k$

    Calculate $\widetilde{\nabla f}(y_t^k, \eta_t^k) = \nabla f(y_t^k) + \eta_t^k$

    Set $x_{t+1}^k = y_t^k - \alpha_{k,t}\widetilde{\nabla f}(y_t^k, \eta_t^k)$

We now proceed to establishing the differential privacy of Algorithm 5 formally by combining the results of Laplace mechanism and composition. The proof follows the same steps as in Proposition 4.2.1 with $\beta = 1$.

Proposition 4.2.6. *The proposed algorithm in* (5) *is ϵ-DP private when the variance of the noise at any iteration t of stage k is*

$$\sigma_{k,t} = \frac{S_1(x_t^k)}{\epsilon/K}, \tag{4.40}$$

where $S_1(x_t^k)$ is the sensitivity bound, K is the total number of stages and n is the data size. By using this approach, the variance of noise at each iteration t, stage k can be computed as

$$\sigma_{k,t} = \frac{S_1(x_t^k)}{\epsilon/(Kn_k)} = \frac{S_1(x_t^k)}{\epsilon_k/(n_k)}, \tag{4.41}$$

where n_k and ϵ_k is the number of iterations and the privacy level at stage $k = 1, 2, ..., K$, respectively.

Determining the Stepsize

Similar to the previous algorithm DP-GDwS, we take into account the variance of $\widetilde{\nabla f}(x_t^k)$ with respect to the noise terms until time t to adjust the stepsize. We then suggest a general form based on the formulation in the original work [3]. Our noise related approach for the stepsize $\alpha_{k,t}$ at stage $k > 1$ and iteration t is

$$\alpha_{k,t} \approx \frac{1}{2^{2(k+1)}L} \left[\text{Var} \left(\widetilde{\nabla f}^{(k,t)} \underbrace{(x_t^k, \ldots, x_t^k)}_{n_k+1 \text{ times}} \right) \right]^{-1/2}, \tag{4.42}$$

where the term in square brackets is again the average over the variances with respect to the components of the gradient vector. By using the Proposition 4.2.2 and the bound presented in Appendix A.2, we choose the stepsize as

$$\alpha_{k,t} = \frac{1}{2^{2(k+1)}2L} \left(\left(\frac{S_2(x_t^k)^2}{4d} + \frac{2S_1(x_t^k)^2}{\left(\frac{\epsilon}{n_k K}\right)^2} \right) \frac{1}{1-\beta^2} \right)^{-1/2}, \quad k = 1, \ldots, K, \tag{4.43}$$

where ϵ and T are the privacy loss and the number of iterations, respectively. Since we compute the sensitivity bounds as independent from the iteration vector x_t^k, the bounds can be taken as S_1 and S_2 and our stepsize formulation becomes independent from t.

Convergence Analysis

Our convergence analysis is based on the dynamical system representation of the algorithm. The linear dynamical system representation of the accelerated gradient descent method at iteration t can be written as

$$\begin{aligned} \xi_{t+1} &= A\xi_t + B\nabla f(y_t, \eta_t), \\ y_t &= C\xi_t, \end{aligned} \tag{4.44}$$

where $\xi_t = \begin{bmatrix} x_t^T & x_{t-1}^T \end{bmatrix}^T \in \mathbb{R}^{2d}$ is the state vector and A, B and C are system matrices with appropriate dimensions defined as Kronecker products $A = \tilde{A} \otimes I_d$, $B = \tilde{B} \otimes I_d$ and $C = \tilde{C} \otimes I_d$ with

$$\tilde{A} = \begin{bmatrix} 1+\beta_{AG} & -\beta_{AG} \\ 1 & 0 \end{bmatrix},\ \tilde{B} = \begin{bmatrix} -\alpha_{AG} \\ 0 \end{bmatrix}, \tilde{C} = \begin{bmatrix} I & 0 \end{bmatrix},\ \tilde{D} = 0,$$

where β_{AG} and α_{AG} are momentum and stepsize parameters for NAG method respectively. Since we are dealing with noisy gradient instead of gradient itself, we will use its noisy version which will be shown as $\widetilde{\nabla f}(y_t, \eta_t)$. From the main theorem related to M-ASG (Theorem 3.4, [3]), the following result is satisfied for DP-MAG.

Theorem 4.2.7. *Let $f \in S_{\mu,L}(\mathbb{R}^d)$. Consider running DP-MAG with the following parameters:*

$$\begin{aligned} &n_1 \geq 0, \quad \alpha_1 = \frac{1}{2^{2(k+1)}L} \\ &n_k = 2^k \left\lceil \sqrt{\kappa}\log(2^{p+2}) \left(\left[\frac{S_2^2}{4d} + \frac{2S_1^2}{\left(\frac{\epsilon}{n_k K}\right)^2} \right] \frac{1}{1-\beta^2} \right)^{1/4} \right\rceil, \\ &\alpha_k = \frac{1}{2^{2(k+1)}2L} \left(\left[\frac{S_2^2}{4d} + \frac{2S_1^2}{\left(\frac{\epsilon}{n_k K}\right)^2} \right] \frac{1}{1-\beta^2} \right)^{-1/2} \end{aligned} \tag{4.45}$$

for any $k > 1$ and $p \geq 1$ and t is the current iteration value. The last iterate of each stage, i.e, $x_{n_k+1}^k$, satisfies the following bound for all $k \geq 1$:

$$\mathbb{E}[f(x_{n_k+1}^k)] - f^* \leq \frac{2}{2^{(p+1)(k-1)}}(\exp(-\frac{1}{8}n_1/\sqrt{\kappa})(f(x_0^0) - f^*)) + \frac{\max(\sigma_k^2)\sqrt{\kappa}}{8L2^{k-1}}. \tag{4.46}$$

Note that as in [3], we take the number of iterations in the first stage as

$$n_1 = \lceil 2\sqrt{\kappa}\log\sqrt{\kappa} \rceil.$$

4.3 Momentum-Based Algorithms Using Sampling

This section is reserved for the stochastic versions of the algorithms presented in the previous chapter. It is known in the literature that the uncertainity coming from the random selection of data helps to protect the privacy. By using this idea, we aim to improve the privacy of DP-GDwS and DP-MAG with the help of sampling. The

following sections explain the details related to the new versions and the parameter selection schemes. Our idea of using noise related stepsize again results in better performance when we compare our results against the standard stepsize formulations.

4.3.1 Stochastic Gradient Descent Algorithm with Smoothing

When privacy is of concern for the optimization problem given in Section 4.1.1, one approach is to update the parameter x_t using a noisy stochastic gradient vector

$$\widetilde{\nabla f}_{B_t}(x_t) = \nabla f_{B_t}(x_t) + \eta_t, \tag{4.47}$$

where B_t is the random sample and the noise term is a Laplace distributed random variable

$$\eta_{t,i} \overset{\text{i.i.d.}}{\sim} \text{Lap}(\sigma_t), \quad i = 1, \ldots, d,$$

independent from the other noise terms and $\eta_t = [\eta_{t,1}, ..., \eta_{t,d}]$. We combine this idea of subsampling the data with our previous algorithms introduced in Section 4.2.1 and Section 4.2.2.

Indeed, the differentially private algorithm in [76] returns the sequence $\{x_t\}_{t\geq 1}$ that is produced as

$$x_{t+1} = x_t - \alpha_t \widetilde{\nabla f}_{B_t}(x_t). \tag{4.48}$$

In [76], α_t is the stepsize at time t and B_t's are disjoint and known to the analyst. If we want this update to be ϵ-DP, then σ_t has to be taken at least $S_1(x_t)/\epsilon$, where for $x \in \mathcal{X}$, $S_1(x)$ is the L_1-sensitivity of the $f(x)$. In [76], since every data sample is used only once in gradient calculation, the overall algorithm is also ϵ-DP. We next employ this idea of subsampling the data and present the online version of DP-GDwS which we call DP-onlineGDwS.

Algorithm DP-SGDwS

In this part, we introduce a stochastic algorithm which uses weighted averages of weighted averages of the current and the previous noisy gradients. More concretely, the update rule for our algorithm is

$$x_{t+1} = x_t - \alpha_t \widetilde{\nabla f}^{(t)}_{B_{1:t}}(x_{1:t}), \tag{4.49}$$

where $\widetilde{\nabla f}^{(t)}_{B_{1:t}}(x_{1:t})$ is the smoothed stocghastic gradient estimate. Specifically, letting

$$\widetilde{\nabla f}^{(0)}_{B_0}(x_0) = \beta \nabla f(x_0) + \eta_0,$$

we define the rest recursively as

$$\widetilde{\nabla f}^{(t)}_{B_{1:t}}(x_{1:t}) = (1-\beta)\widetilde{\nabla f}^{(t-1)}_{B_{1:t-1}}(x_{1:t-1}) + \beta \nabla f_{B_{1:t}}(x_t) + \eta_t. \tag{4.50}$$

With the help of smoothing approach, the new information that an adversary can learn from the x_t given the previous samples $x_1, \ldots, x_{t-1}$ becomes

$$\beta \nabla f_{B_t}(x_t) + \eta_t, \tag{4.51}$$

which is a noise-added calculation. Again, for $\nabla f_{B_t}(x_t) + \eta_t$, the sensitivity of the calculation is $1/\beta$ times more and therefore the privacy breach is $1/\beta$ times more.

Algorithm 6 describes our stochastic gradient descent method for solving the generic problem formulated in (4.1) when the desired privacy level is ϵ and the maximum number of iterations is T. We note that a similar strategy to ours that combines mini-batching with a noise-adding mechanism for averaged quantities has been used in [64]; however in a different setting for the purpose of private variational Bayesian inference.

We now proceed to establishing the differential privacy of Algorithm 6 formally. The next result is simply obtained by combining the results of Laplace mechanism, subsampling and composition.

Proposition 4.3.1. *The proposed smoothing scheme in* (4.15) *leads to an ϵ-DP private algorithm ($\epsilon \leq 1$) when the variance of the noise at any iteration t is*

$$\sigma_t = \frac{S_1(x_t)\beta}{\frac{n\epsilon}{2mT}}, \tag{4.52}$$

where $S_1(x_t)$ is the sensitivity bound, β is the smoothing weight, m is the subsample size, T is the maximum number of iterations and n is the data size.

Proof. Assume B_t is fixed and known. Observing (4.3), we see that change in one data item changes only one term in $f_{B_t}(x_t)$. Therefore, the sensitivity of $f_{B_t}(x_t)$ is equal to the sensitivity of $f(x_t)$, which we denoted to be $S_1(x_t)$. Hence, if B_t were known, the privacy loss of revealing (4.51) with σ_t in (4.52) the at iteration t would be

$$\epsilon_0 = S_1(x_t)\beta/\sigma_t = \frac{n\epsilon}{2mT}.$$

Algorithm 6: DP-SGDwS: Differentially private smoothed stochastic gradient descent algorithm

Input: Sensitive data $y_{1:n}$, initial value x_0, subsample size m, number of iterations T, total privacy breach ϵ

Output: The adversar's view: x_t, $t = 1, \ldots, T$.

for $t = 0, \ldots, T-1$ **do**

  Sample B_t of size m uniformly from $\{1, \ldots, n\}$.

  Calculate $\nabla f_{B_t}(x_t) = \sum_{i \in B_t} \nabla f_i(x_t)$

  Calculate $S_1(x_t)$ and set $\sigma_t = \frac{S_1(x_t)\beta}{\frac{n\epsilon}{2mT}}$.

  Sample $\eta_t \sim \text{Lap}(\sigma_t)$

  if $t = 0$ **then**

    Initialise

$$\widetilde{\nabla f}^{(0)}_{B_0}(x_0) = \beta \nabla f_{B_0}(x_0) + \eta_0.$$

  else

    Calculate

$$\widetilde{\nabla f}^{(t)}_{B_{1:t}}(x_{1:t}) = (1-\beta)\widetilde{\nabla f}^{(t-1)}_{B_{1:t-1}}(x_{1:t-1}) + \beta \nabla f_{B_t}(x_t) + \eta_t.$$

  Update $x_{t+1} = x_t - \alpha_t \widetilde{\nabla f}^{(t)}_{B_{1:t}}(x_{1:t})$.

By Theorem 4.1.5, subsampling with size m out of n samples makes the privacy per iteration equal to $2\epsilon_0 m/n = \epsilon/T$. Finally, we apply Theorem 4.1.3 to conclude that the privacy loss after T iterations is ϵ. □

Determining the Stepsize

In order to adjust the stepsize for the differential privacy, we look at the variance of $\widetilde{\nabla} f(x_t)$ with respect to the joint distribution of $(B_{1:t}, \eta_{1:t})$, the subsamples and the noise terms until time t. Similar to the previous section, we then suggest the following general form for the stepsize α_t

$$\alpha_t \approx \left[\mathrm{Var}\left(\widetilde{\nabla} f^{(t)}_{B_{0:t}} \underbrace{(x_t, \ldots, x_t)}_{t+1 \text{ times}} \right) \right]^{-1/2} \frac{1}{(T+1)^\gamma}, \tag{4.53}$$

where T is the number of iterations and $\gamma \in (0, 1]$ is constant. We present the following bound on the variances; the proof is in Appendix A.2.

Proposition 4.3.2. *For given $x_t \in \mathcal{X}$, subsample size m, data size n, smoothing parameter β where $0 < \beta \leq 1$, and $\eta_i \overset{\text{i.i.d.}}{\sim} \mathrm{Lap}(\sigma)$ for $i = 1, \ldots, t$, the trace of the covariance of $\widetilde{\nabla} f^{(t)}_{B_{1:t}}(x_t, \ldots, x_t)$ with respect to the joint distribution of $(B_{1:t}, \eta_{1:t})$ is bounded as*

$$\mathrm{Var}\left(\widetilde{\nabla} f^{(t)}_{B_{0:t}} \underbrace{(x_t, \ldots, x_t)}_{t+1 \text{ times}} \right) \leq \left[\frac{S_2(x_t)^2}{4} m \left(1 - \frac{m-1}{n-1} \right) + 2d\sigma^2 \right] \frac{\beta}{2-\beta}. \tag{4.54}$$

In the light of the above proposition, in Algorithm 6, if the privacy loss and the number of iterations are chosen as ϵ and T, respectively, we choose the stepsize as follows:

$$\alpha_t = \left(\left[\frac{S_2(x_t)^2}{4d} m \left(1 - \frac{m-1}{n-1} \right) + \frac{2S_1(x_t)^2}{\left(\frac{n\epsilon}{2mT}\right)^2} \right] \frac{\beta}{2-\beta} \right)^{-1/2} \frac{0.25}{(T+1)^{1/2}}. \tag{4.55}$$

Note that we have T, the maximum number of iterations, in the formulation of stepsize. Since T is fixed and we can compute the sensitivity bounds S_1 and S_2 as independent from x_t, our stepsize formulation is constant during iterations similar to previous section.

4.3.2 Multistage Accelerated Stochastic Algorithm

In this part, we present the stochastic version of DP-MAG algorithm which is presented in Section 4.2.2. As it is mentioned before, the amplification effect of sampling is one of the easy ways to improve the differential privacy. With our main aim of obtaining private algorithms with less noise and better performance, we will present the stochastic version of DP-MAG, which we denote by DP-SMAG.

Algorithm DP-SMAG

The pseudocode of DP-SMAG is given in Algorithm 7. The only difference with Algorithm 5 is the sampling step in Line 8. Again, to determine the noise variance, we first define the privacy level per stage and then the privacy level per iteration. Different from DP-MAG, we add the related noise to the gradient that is computed with respect to the subsampled data.

Algorithm 7: DP-SMAG: Differentially private stochastic multistage accelerated algorithm

1 **Input:** Sensitive data $y_{1:n}$, initial value x_0^0, the sequence of stepsizes $\{\alpha_k\}_{k=1}^K$, number of iterations at each stage $\{n_k\}_{k=1}^K$, total privacy breach ϵ

2 **Output:** The adversar's view: x_t, $t = 1, \dots, T$.

3 Set $n_0 = -1$

4 **for** $k = 1; k \leq K; k = k + 1$ **do**

5 Compute $\epsilon_k = \epsilon / K$

6 Set $x_0^k = x_1^k = x_{n_{k-1}+1}^{k-1}$

7 **for** $t = 1; t \leq n_k; t = t + 1$ **do**

8 Sample $B_{k,t}$ of size m uniformly from $\{1, \dots, n\}$.

9 Calculate $S_1(x_t^k)$ and set $\sigma_{k,t} = \frac{S_1(x_t^k)}{\frac{n\epsilon_k}{2mn_k}}$.

10 Sample $\eta_t^k \sim \text{Lap}(\sigma_{k,t})$

11 Set $\beta_k = \frac{1-\sqrt{\mu/L}}{1+\sqrt{\mu/L}}$

12 Set $y_t^k = (1 + \beta_k)x_t^k - \beta_k x_{t-1}^k$

13 Calculate $\widetilde{\nabla} f_{B_{k,t}}(y_t^k, \eta_t^k) = \nabla f_{B_{k,t}}(y_t^k) + \eta_t^k$

14 Set $x_{t+1}^k = y_t^k - \alpha_{k,t}\widetilde{\nabla} f_{B_{k,t}}(y_t^k, \eta_t^k)$

The following proposition proves the differential privacy of Algorithm 7 formally by combining the results of Laplace mechanism and composition. The proof follows the same steps as the proof of Proposition 4.3.1 with $\beta = 1$ and hence, we skip the proof.

Proposition 4.3.3. *The proposed algorithm in* (7) *with sample size* m *is* ϵ*-DP private when the variance of the noise at any stage* k *is*

$$\sigma_{k,t} = \frac{S_1(x_t^k)}{\epsilon/K}, \tag{4.56}$$

where $S_1(x_t^k)$ *is the sensitivity bound,* K *is the total number of stages, and* n *is the data size. By using this approach, the variance of noise at each iteration* t*, stage* k *can be computed as*

$$\sigma_{k,t} = \frac{S_1(x_t^k)}{\frac{n\epsilon}{2mKn_k}} = \frac{S_1(x_t^k)}{\frac{n\epsilon_k}{2mn_k}}, \tag{4.57}$$

where ϵ_k *is the privacy level and* n_k *is the number of iterations at stage* $k = 1, 2, ..., K$*.*

Determining the Stepsize

Similar to the previous algorithm DP-SGDwS, to adjust the proportionality constant of the stepsize, we look at the variance of $\widetilde{\nabla} f_{B_t}(x_t^k)$ with respect to the noise terms until time t. We then suggest the following general form based on the formula in the original work [3] and our noise related approach for the stepsize $\alpha_{k,t}$ at stage $k > 1$ and iteration t is

$$\alpha_{k,t} \approx \frac{1}{2^{2(k+1)}2L} \left[\text{Var} \left(\left\| \widetilde{\nabla} f^{(t)}_{B_{k,t}} \underbrace{(x_t^k, \ldots, x_t^k)}_{n_k+1 \text{ times}} \right\|_2^2 \right) \right]^{-1/2}. \tag{4.58}$$

The stepsize formulation is depend on both k and t. However, since we can compute upper bounds S_1 and S_2 as independent from the iteration vectors, the dependence over t is removed. Thus, when the privacy loss and the number of iterations are chosen as ϵ and T, respectively, we choose the following stepsize in Algorithm 7

$$\alpha_k = \frac{1}{2^{2(k+1)}2L} \left(\left[\frac{S_2^2}{4d} m \left(1 - \frac{m-1}{n-1} \right) + \frac{2S_1^2}{\left(\frac{n\epsilon}{2mn_kK} \right)^2} \right] \frac{1}{1-\beta^2} \right)^{-1/2}, \quad k = 1, \ldots, K. \tag{4.59}$$

Similar to the DP-MAG algorithm, the number of iterations at stage $k \geq 1$ is computed as

$$n_k = 2^k \left\lceil \sqrt{\kappa} \log(2^{p+2}) \left(\left[\frac{S_2^2}{4d} m \left(1 - \frac{m-1}{n-1} \right) + \frac{2S_1^2}{\left(\frac{n\epsilon}{2mn_k K} \right)^2} \right] \frac{1}{1-\beta^2} \right)^{1/4} \right\rceil .$$

4.4 Computational Study

All simulation experiments in this section are performed for a regularized logistic regression problem for binary classification. The model has observations $y_i = (\zeta_i, z_i)$, $i = 1, \ldots, n$, where $\zeta_i \in \mathcal{Z} \subseteq \mathbb{R}^d$ is a $d \times 1$ vector of covariates and $z_i \in \{-1, 1\}$ is a binary response whose conditional probability given ζ_i depends on a $d \times 1$ parameter vector $x \in \mathcal{X} \subseteq \mathbb{R}^d$ as follows:

$$p(z_i|\zeta_i, x) = \frac{1}{1 + e^{-z_i \zeta_i \cdot x}}, \quad i = 1, \ldots, n.$$

Since the probability distribution of ζ_i's does not depend on x, the maximum likelihood problem is defined as in

$$x^* = \arg\max_{x \in \mathcal{X}} \sum_{i=1}^{n} f(x; \zeta_i, z_i), \tag{4.60}$$

where

$$f(x; \zeta_i, z_i) := -\log\left(1 + e^{-z_i \zeta_i \cdot x}\right) + \lambda \|x\|^2.$$

In order to calculate the sensitivity of the gradient of the log-likelihood, observe that for a fixed x, for all ζ, ζ', z, z', we have

$$\begin{aligned} \|\nabla f(x; \zeta, z) - \nabla f(x; \zeta', z')\|_1 &= \left\| \frac{z\zeta \exp(z\zeta \cdot x)}{1 + \exp(z\zeta \cdot x)} - \frac{z'\zeta' \exp(z\zeta' \cdot x)}{1 + \exp(z'\zeta' \cdot x)} \right\|_1 \\ &\leq \|\zeta\|_1 \left| \frac{z\zeta \exp(z\zeta \cdot x)}{1 + \exp(z\zeta \cdot x)} \right| + \|\zeta'\|_1 \left| \frac{z'\zeta'^T \exp(z\zeta' \cdot x)}{1 + \exp(z'\zeta' \cdot x)} \right| \\ &\leq \|\zeta\|_1 + \|\zeta'\|_1 \end{aligned} \tag{4.61}$$

Therefore, $S_1(x) = 2 \sup_{\zeta \in \mathcal{Z}} \|\zeta\|_1$ for all x. Moreover, by following the similar steps one can show that $S_2(x) = 2 \sup_{\zeta \in \mathcal{Z}} \|\zeta\|_2$. For the first part of experiments to follow, we use a synthetic data with $d = 10$ and $n = 10^4$. and the value of regularization parameter λ is taken as 0.01. Although we randomly produce the data, we have tried different

random datasets and the improved performance of our algorithms is still satisfied. Unless otherwise stated, the dataset used in the experiments is synthetic.

The second part of experiments is reserved for the solution of regularized logistic regression problem on MNIST dataset. MNIST contains $60,000 \times 400$ data points and $60,000$ labels. Each labels contains numbers $1, 2, ..., 8$, but we will follow a similar approach to [76] and classify them as 1 vs others (-1).

To determine the noise variance, we need the norm of the gradient which depends on the norm of the dataset as it can be seen in (4.4). That means, the norm of the data highly affects variance of the noise, i.e, the performance of the algorithms. To decrease this effect, we take help of the norm clipping approach which is used in differential privacy context before [1]. This approach works as follows: if the norm of the gradient is less than a threshold constant C, it remains the same. Otherwise, we scale it down to the value C. In our experiments we take this value as 2.

The stepsize for GD (SGD) is taken as $\frac{1}{\sqrt{t}}$ where t is the current iteration as in [76]. For HB (SHB) and NAG (SNAG), the stepsize and momentum parameter $\bar{\beta}$ (which corresponds to $1-\beta$ for DP-GDws and DP-SGDwS) selection is as follows:

$$\alpha_{HB} = \frac{1}{L}, \quad \bar{\beta}_{HB} = \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2,$$

$$\alpha_{NAG} = \frac{1}{L}, \quad \bar{\beta}_{NAG} = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1},$$

respectively.

Now, we proceed to show the performance results for each algorithm in related sections and compare with some existing methods from the literature.

4.4.1 Results for Deterministic Algorithms

In this section, we present results for our deterministic algorithms, DP-GDwS and DP-MAG. All experiments are repeated 10 times and the averages of the results are plotted.

DP-GDwS

The first set of experiments show the results for DP-GDwS. As it is mentioned before, this algorithm aims to obtain an improved performance and privacy by decreasing the

noise variance. The smoothing scheme, which is based on taking the coefficient of the current gradient is $\beta \in (0, 1)$ instead of 1, helps to decrease the variance of the noise. Thus, we first compare the resulting plots for various β values at various privacy levels in Figure 4.2 to demonstrate the effect of smoothing. The x-axis and y-axis represents the number of iterations and the objective function error, respectively. Most of the plots in this chapter are constructed by using logarithmic scaling of the axes. However, since the performance differences for different β values are seen more clearly, we do not use logarithmic scaling for these figures.

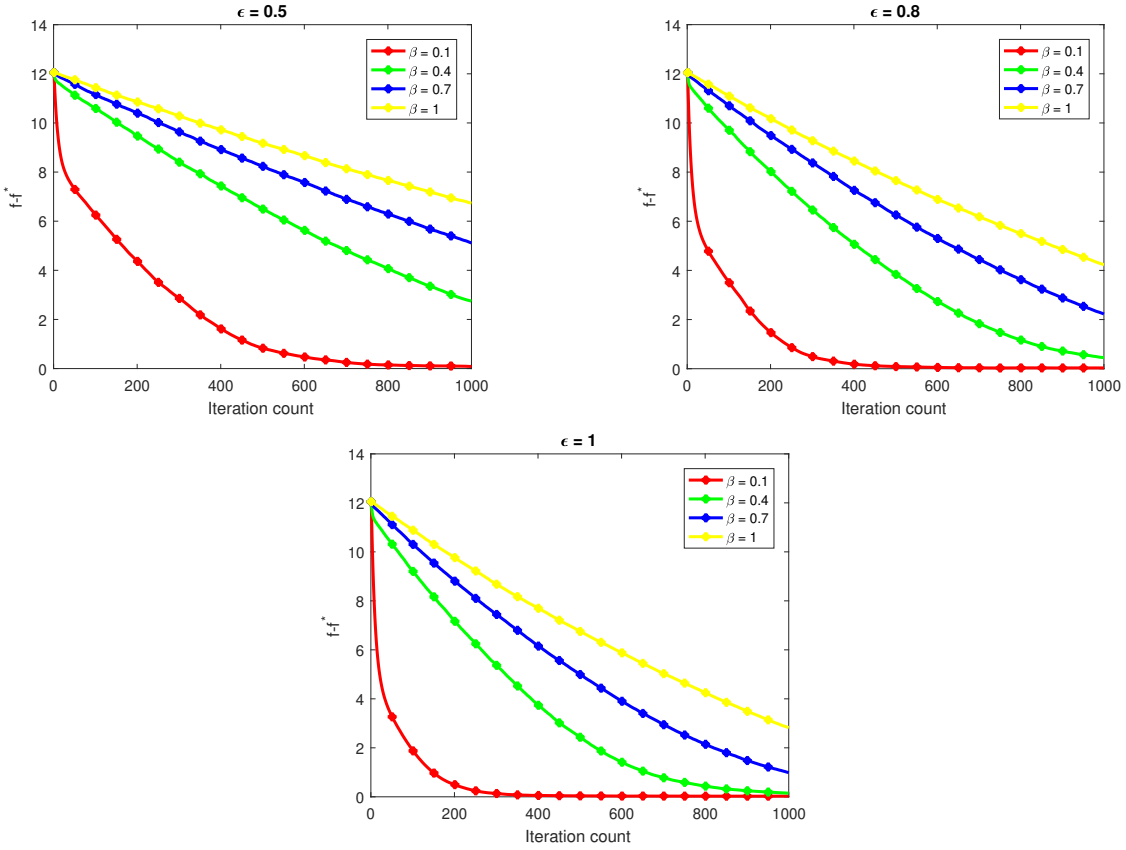


Figure 4.2: Results of DP-GDwS for $\epsilon = 0.5, 0.8, 1$

Figure 4.2 shows that the decreasing β values results in better performance for various privacy levels. This is an important result, since we achieve to decrease the objective function error and make the algorithm satisfy the same privacy level with less noise. Moreover, the results show that our approach achieve the best performance even at tighter privacy levels. The case $\beta = 1$ corresponds to the differentially private gradient descent algorithm (DP-GD) with noise related stepsize and our smoothing approach beats this version of DP-GD. To show our contribution more clearly, we plot

the final value of the objective error, $f - f^*$, vs $\beta \in \{0.01, 0.02, ..., 0.99, 1\}$ values for different privacy levels in Figure 4.3. It is clearly seen that increasing beta values result in higher objective function error and the best performance is obtained around $\beta = 0.1$. In other words, we achieve the aim of performing better while satisfying the same privacy level with less noise.

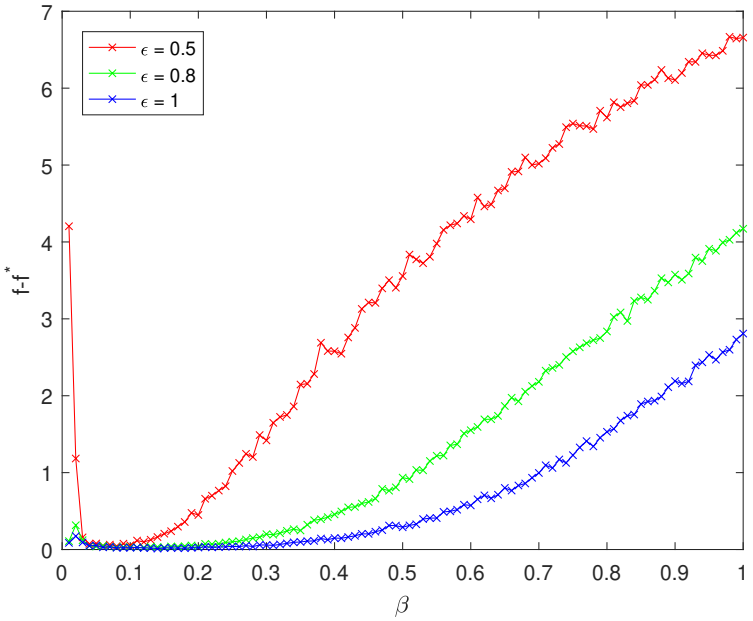


Figure 4.3: Results of DP-GDwS for $\epsilon = 0.5, 0.8, 1$

DP-MAG

This part is reserved for the performance of our second algorithm DP-MAG which is based on the existing M-ASG algorithm. Although M-ASG uses the noisy gradients to take steps, the amount of noise is not determined by considering the differential privacy. To show that our adjustments result in an improved performance in DP, we first give a comparison figure for DP-MAG and differentially private version of M-ASG. As it is shown in Figure 4.4, for various privacy levels, our version performs much better than M-ASG and M-ASG* which is a variant of M-ASG [3]. In other words, our approach, which is using noise related stepsize along with a special noise dividing scheme, helps to improve the algorithm under the amount of noise required for differential privacy. Note that all of the experiments in this section are run for 10^3 iterations, however, to clarify that M-ASG and M-ASG* cannot catch up with DP-MAG when the number of

iterations increases, we provide another figure, Figure 4.5. The number of iterations is chosen as 10^4 for this figure. The resulting plots show that DP-MAG has still better performance than others.

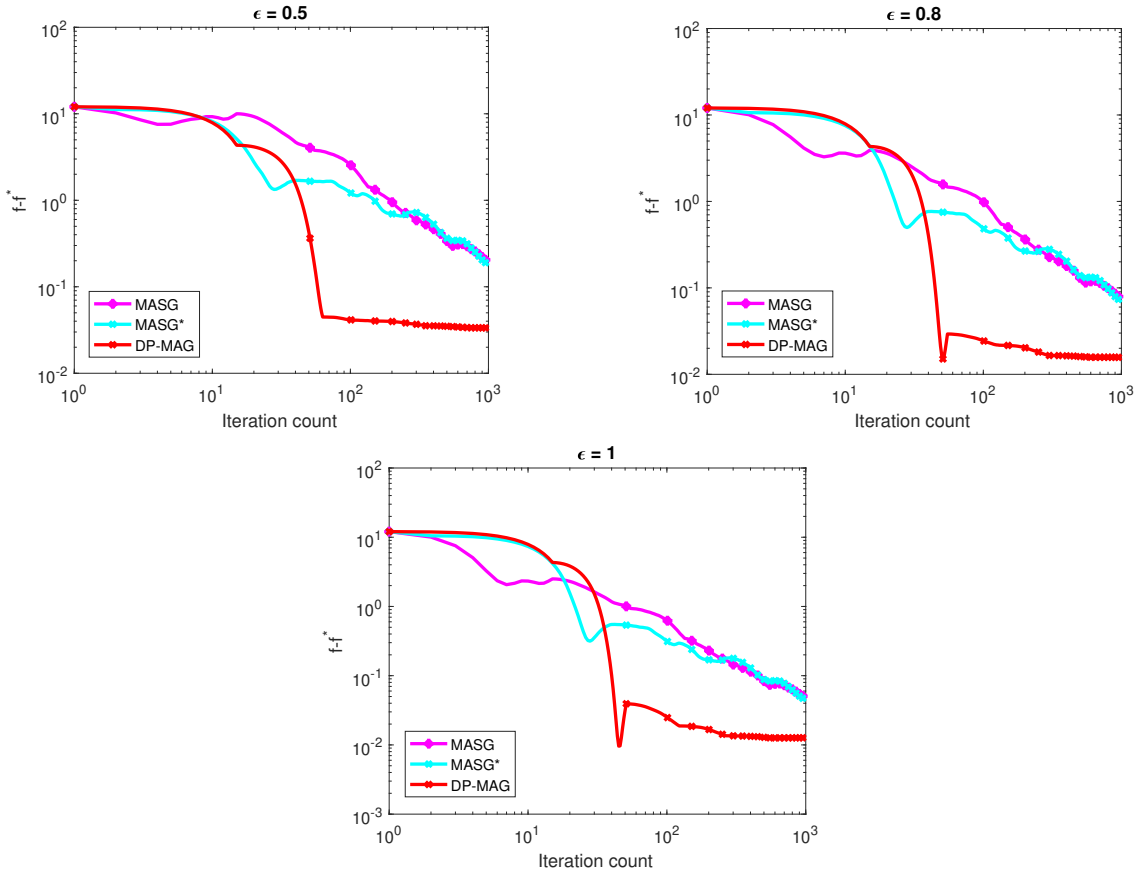


Figure 4.4: Advantage of DP-MAG

Comparison of Differentially Private Deterministic Algorithms

Next, we proceed to compare the performance of our methods DP-GDwS and DP-MAG against differentially private version of other well-known algorithms. The results can be found in Figure 4.6. Since the smoothing achieves improvement over the performance, we take $\beta = 0.1$ for DP-GDwS. It is clearly seen from Figure 4.6 that our algorithms perform better than the differentially private versions of GD, NAG, HB, M-ASG and M-ASG*. We again note that M-ASG* is another multistage accelerated algorithm defined in [3] with noisy gradient.

In the second part, we compare the performance of the algorithms for different datasets. As it is mentioned, we present the results for a synthetic dataset, however,

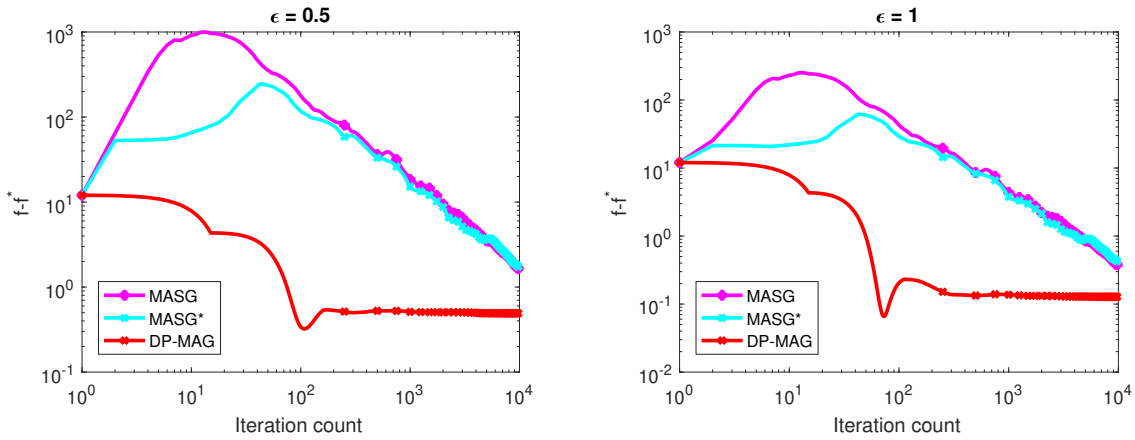


Figure 4.5: Advantage of DP-MAG for 10^4 iterations

our experiments show that reproducing the data with different seed values does not change the result; DP-GDwS and DP-MAG still performs better than the other private algorithms. To prove that, we give the resulting figure in Figure 4.7. This plot shows the final value of the objective error $f - f^*$ of all algorithms for ten different dataset and $\epsilon = 0.5$. The results for DP-NAG and DP-HB are not included since their objective error is high and affect the scaling of the y-axis. The figure shows that our algorithms clearly perform better than the others and it is independent from the random dataset.

4.4.2 Results for Stochastic Algorithms

We now present the numerical results for our stochastic algorithms with sample sizes $1, 10, 100$ and 1000. Each experiment is performed 10 times and the averages of related results are given.

DP-SGDwS

This section includes the result of our first stochastic algorithm DP-SGDwS. Similar to DP-GDwS, our aim for this algorithm is to obtain better performance by keeping the same privacy level. So, to check whether we achieve this purpose, we first give results for various β values and privacy levels. Again, $\beta = 1$ corresponds to DP-SGD with noise related stepsize. The resulting plots can be found in Figures 4.8, 4.9, 4.10 and 4.11 for sample sizes 1, 10, 100 and 1000, respectively. These results show that smoothing approach results in less objective function error while satisfying the same privacy level. Especially for small ϵ values, which means the algorithm is more private,

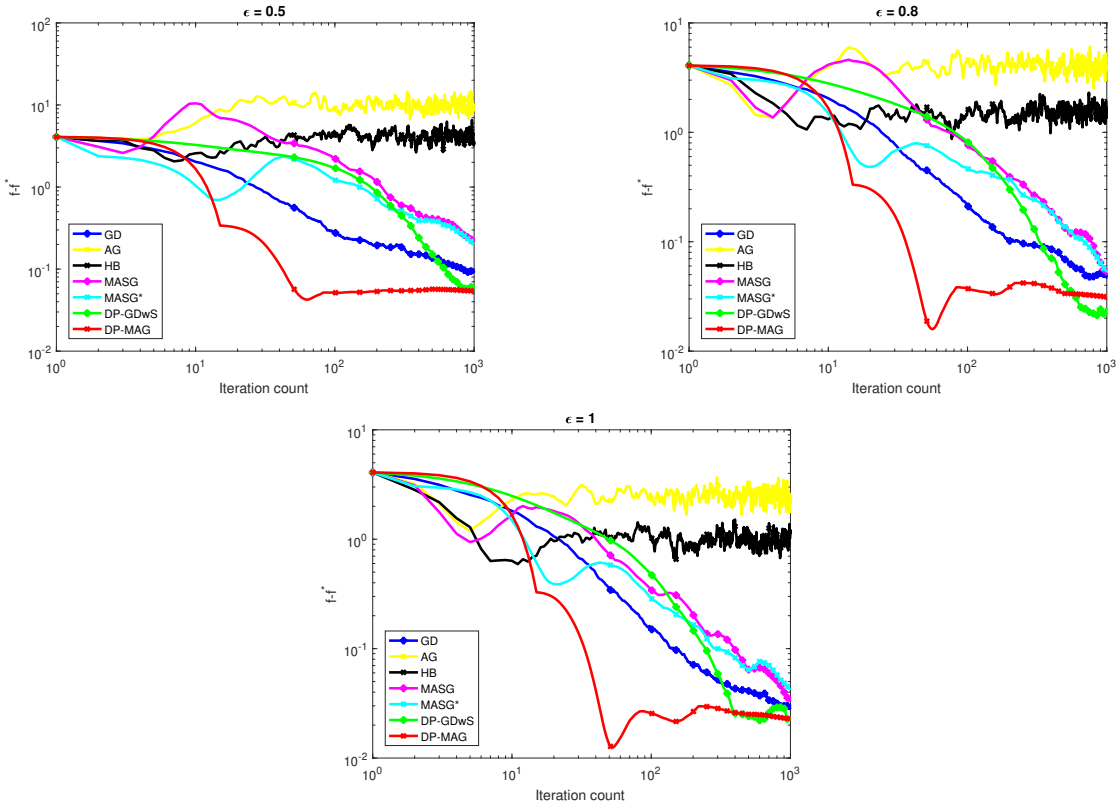


Figure 4.6: Deterministic DP - Comparisons

$\beta = 0.1$ still performs better than the other methods.

Similar to the deterministic version, we again give a comparison of final objective function value for $\beta \in \{0.01, 0.02, ..., 0.99, 1\}$ in stochastic setting. The resulting plot can be found in Figure 4.12. It is clearly seen that our approach of using smaller β result in better performance while satisfying a given privacy level with less noise.

As it is mentioned before, we can also construct an online version of DP-GDwS (DP-onlineGDwS). This algorithm is based on a deterministic setting but since we use the data in subsets, the results are listed in this section. DP-onlineGDwS works as follows: at each iteration, instead of random sampling, we use the data in buckets and the buckets are disjoint at each iteration. The total number of iterations is not predefined as in the other experiments, but the algorithm continues until all data is used once. In other words, if the size of buckets b and data size n, then the number of iterations becomes $\frac{n}{b}$. Figure 4.13 shows the performance of DP-onlineGDwS for bucket size value 10. Again, similar to deterministic and stochastic versions, the smoothing approach helps to improve the algorithm in online setting as well.

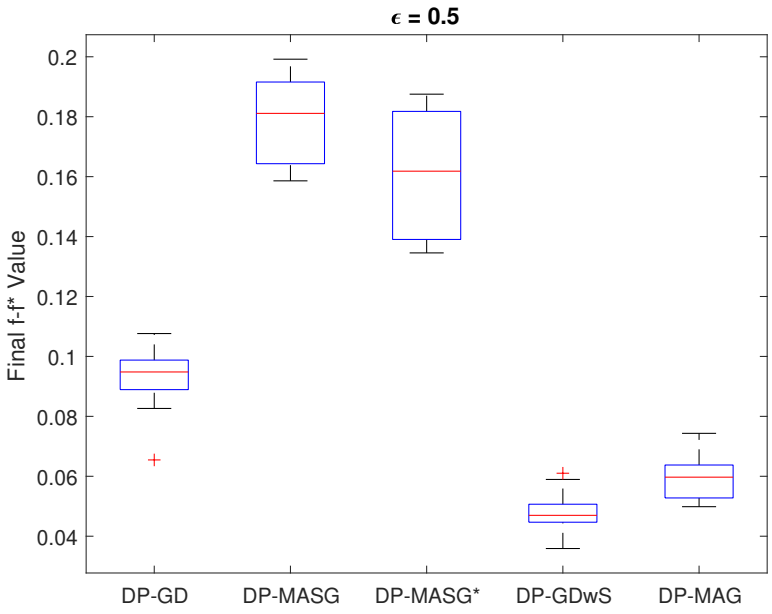


Figure 4.7: Deterministic DP - Comparisons for Different Datasets

DP-SMAG

Now, we present the results for the DP-SMAG algorithm. DP-SMAG is based on the differentially private version of stochastic M-ASG. In other words, we use M-ASG with subsampling, but the algorithm parameters are defined by considering the noise and a special noise dividing scheme is employed. By directly adjusting the amount of noise with respect to the differential privacy, stochastic version of M-ASG and M-ASG* does not perform better than DP-SMAG. So, to show that our approach of noise related stepsize and special noise dividing scheme helps to obtain a better result with stochastic gradient, we compare M-ASG, M-ASG* and DP-SMAG in Figure 4.14. These results are taken for various sample size values and for $\epsilon = 1$. As it is clearly seen, DP-SMAG achieves the best perfomance among these algorithms.

To show that stochastic version of M-ASG does not perform better than our version with the increasing number of iterations, we compare the algorithms with the same parameters and 10^4 iterations. Figure 4.15 shows that with increasing number of iterations (in other words, with increasing noise variance), our algorithm DP-SMAG still performs better than the original versions.

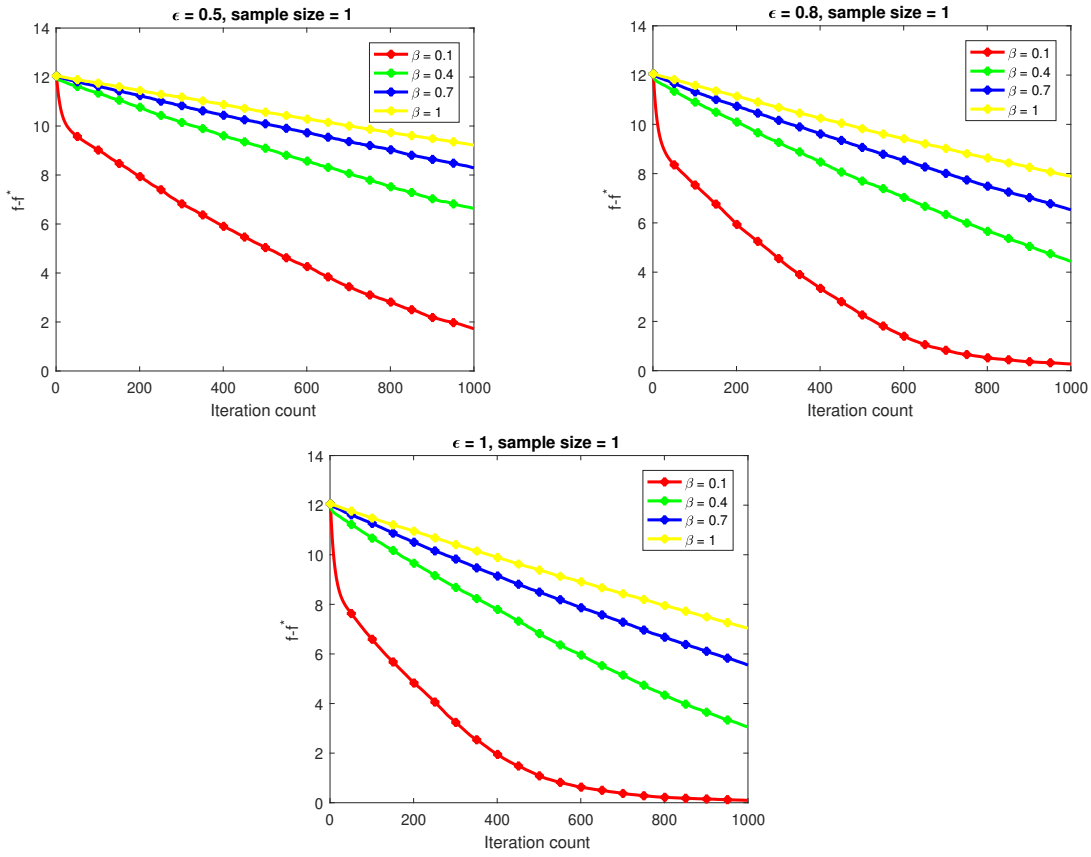


Figure 4.8: DP-SGDwS Results for Sample Size = 1

Comparison of Differentially Private Stochastic Algorithms

This part is reserved for the comparison of DP-SGDwS and DP-SMAG with other differentially private stochastic algorithms SGD, stochastic version of NAG (SNAG) and HB (SHB), stocMASG and stocMASG*. The related results are given in figures 4.16, 4.17, 4.18 and 4.19 for sample sizes 1, 10, 100 and 1000, respectively. Here, β is taken as 0.1 for DP-SGDwS as in the deterministic comparisons plots. It is seen that our algorithms perform better than the other differentially private stochastic algorithms for various privacy levels and sample sizes.

In the second part of the comparisons, we check the performance differences of stochastic algorithms for different synthetic datasets. The resulting plots which show the final $f - f^*$ value for sample sizes 1, 10, 100 and 1000 and $\epsilon = 0.5$ are given in Figure 4.20. It is clearly seen that DP-SGDwS and DP-SMAG performs better than the other algorithms without depending on the dataset.

In the last figures, we discuss the performance of stochastic algorithms for various

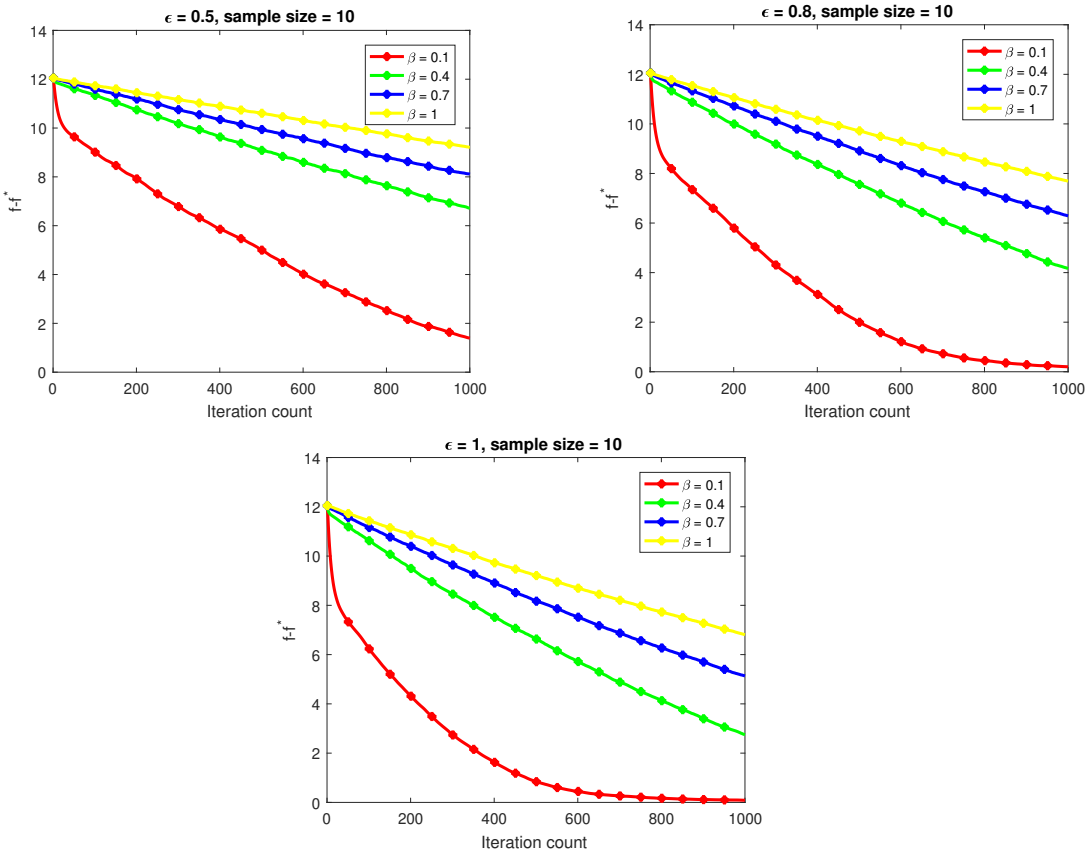


Figure 4.9: DP-SGDwS Results for Sample Size = 10

sample sizes on MNIST dataset. The results for sample sizes 1, 10, 100 and 1000 are given in Figures 4.21, 4.22, 4.23 and 4.24, respectively. These plots show that the best performing algorithm in this setting is DP-SMAG. Although DP-SGDwS is the second best for most of the results, it cannot always overperform the DP-SMAG and stochastic versions of M-ASG and M-ASG*. However, it is clearly seen that smoothing gives a better performance than SGD, NAG and HB for all given privacy levels and sample size values.

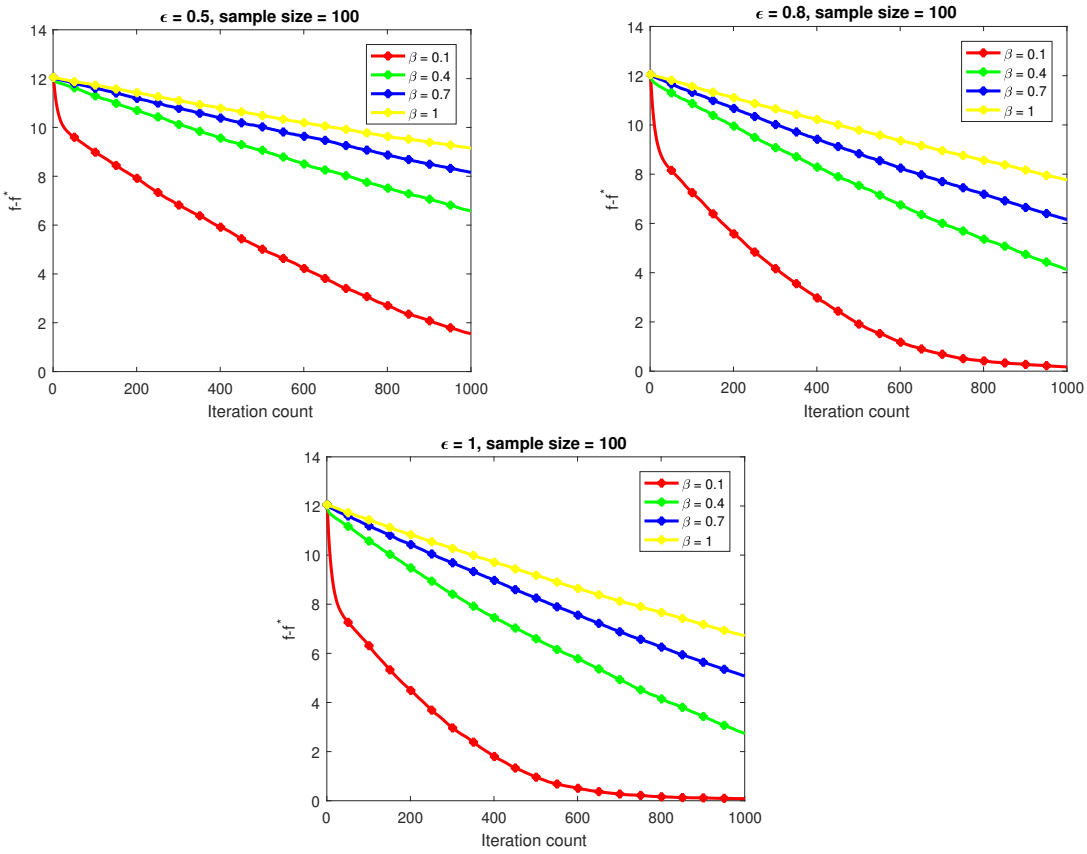


Figure 4.10: DP-SGDwS Results for Sample Size = 100

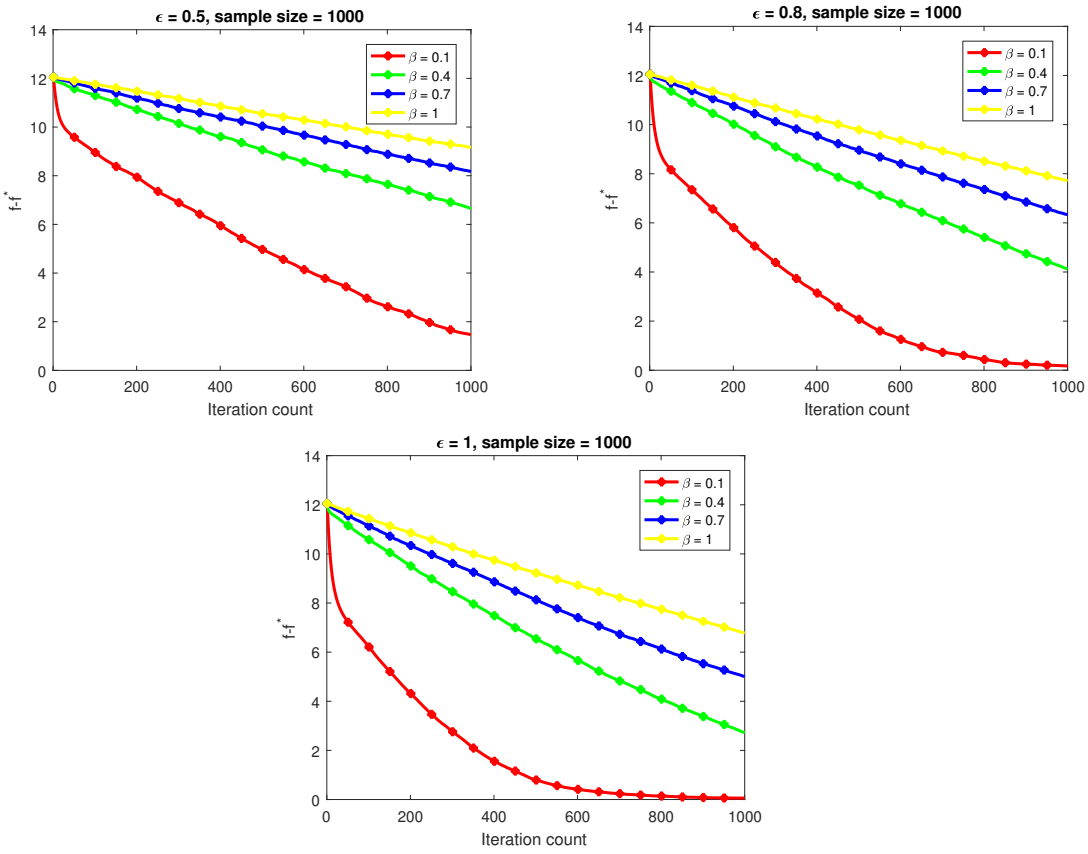


Figure 4.11: DP-SGDwS Results for Sample Size = 1000

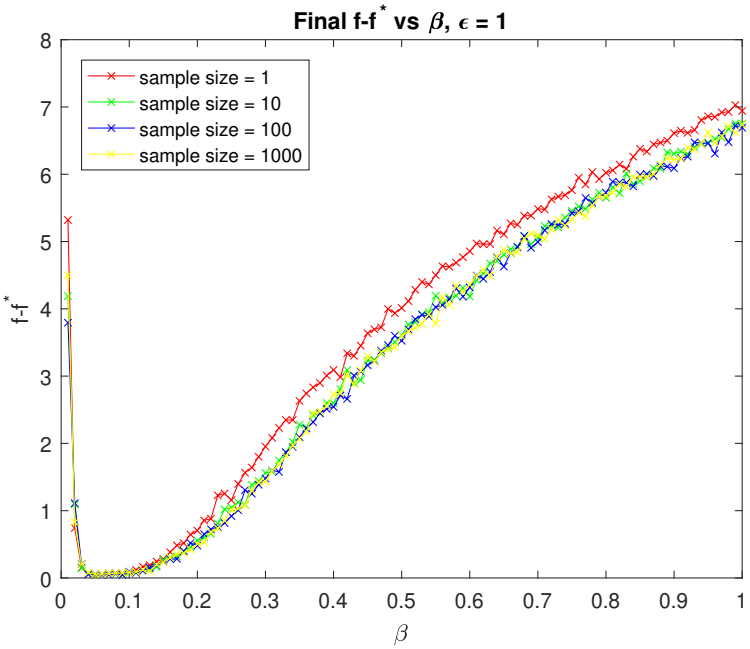


Figure 4.12: Results of DP-SGDwS for $\epsilon = 1$

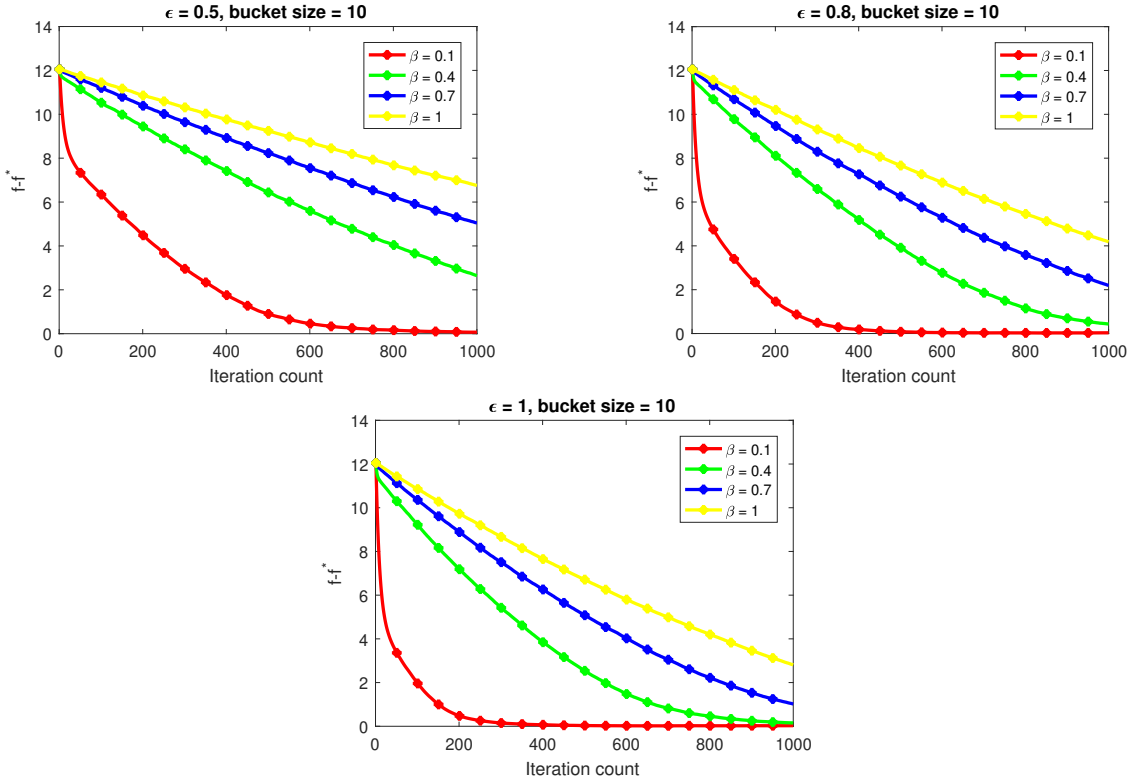


Figure 4.13: DP-onlineGDwS Results for Bucket Size = 10

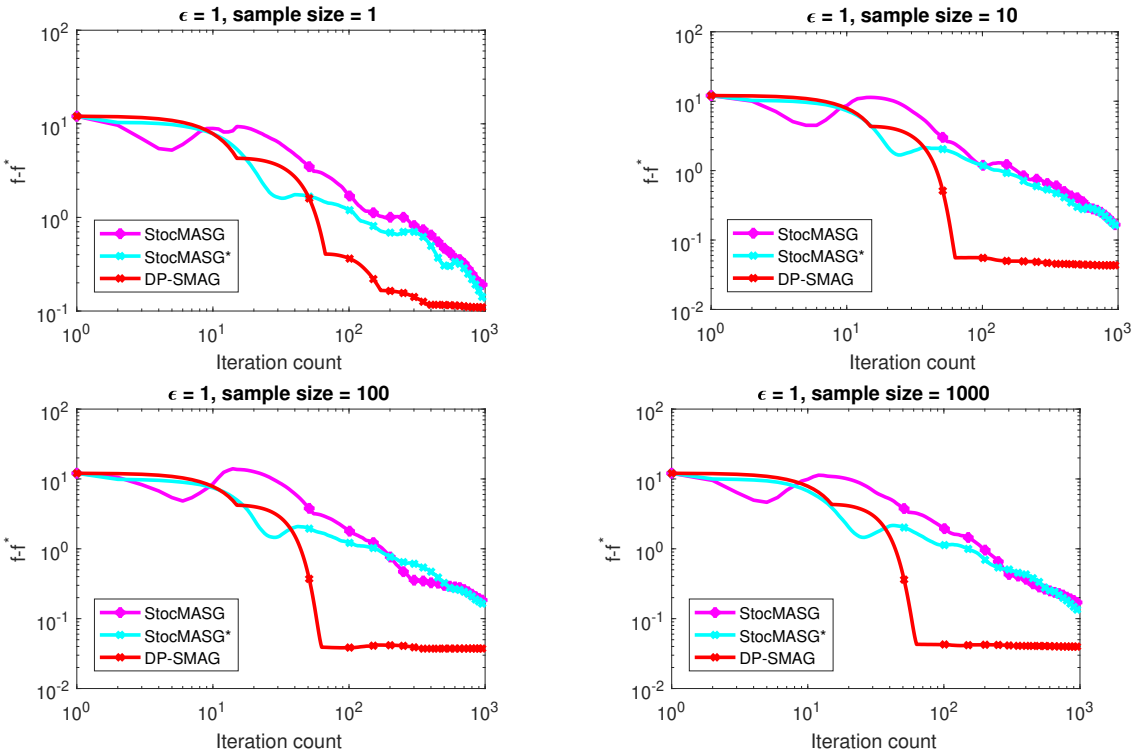


Figure 4.14: Advantage of DP-SMAG For Epsilon = 1

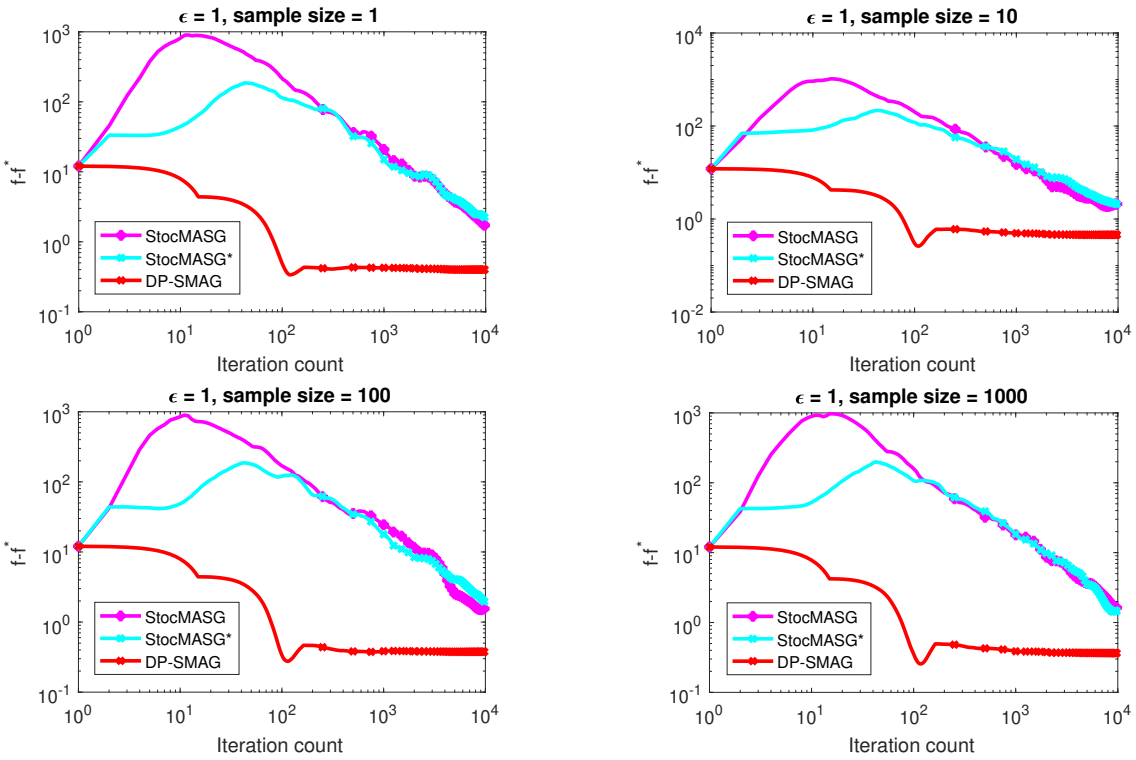


Figure 4.15: Advantage of DP-SMAG For 10^4 Iterations

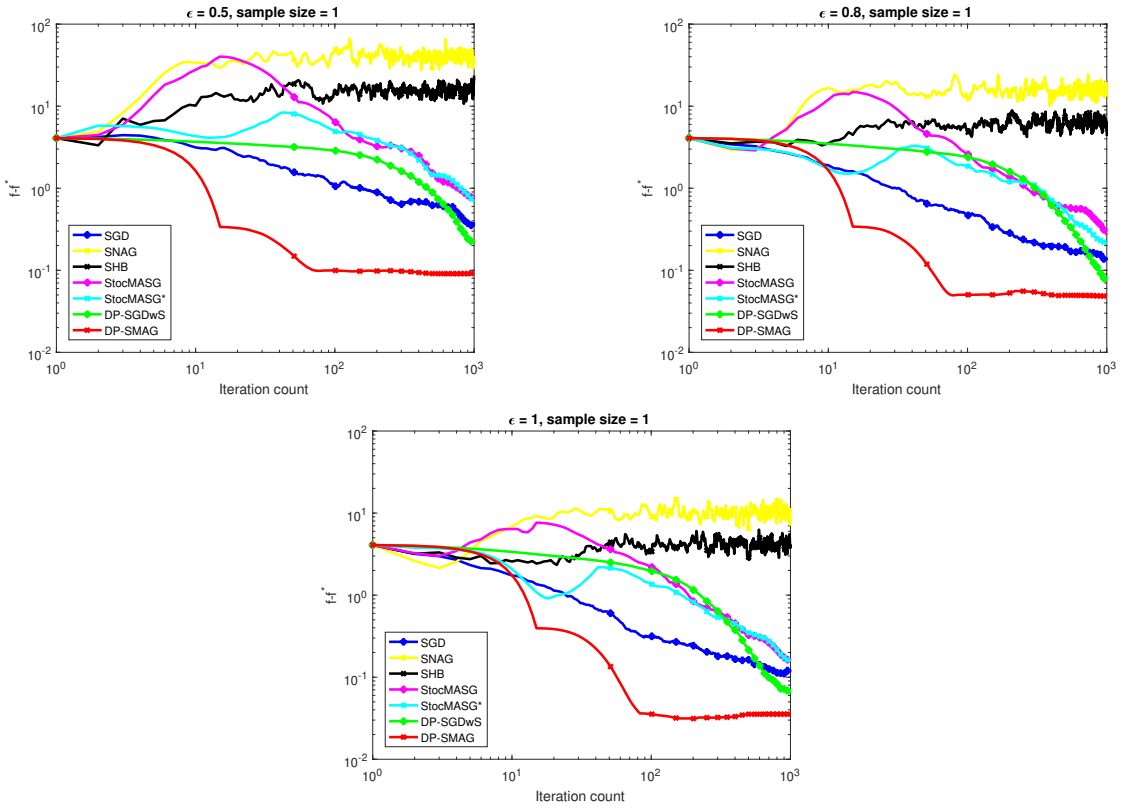


Figure 4.16: Comparison of Stochastic Algorithms For Sample Size = 1

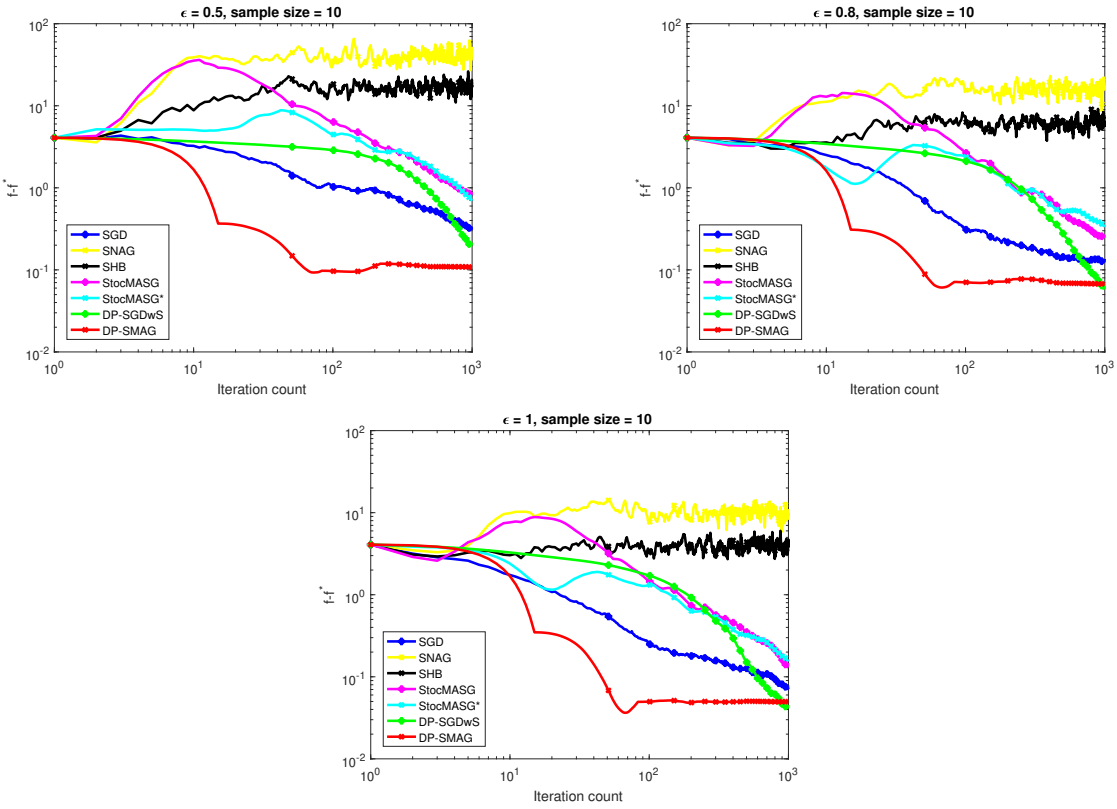


Figure 4.17: Comparison of Stochastic Algorithms For Sample Size = 10

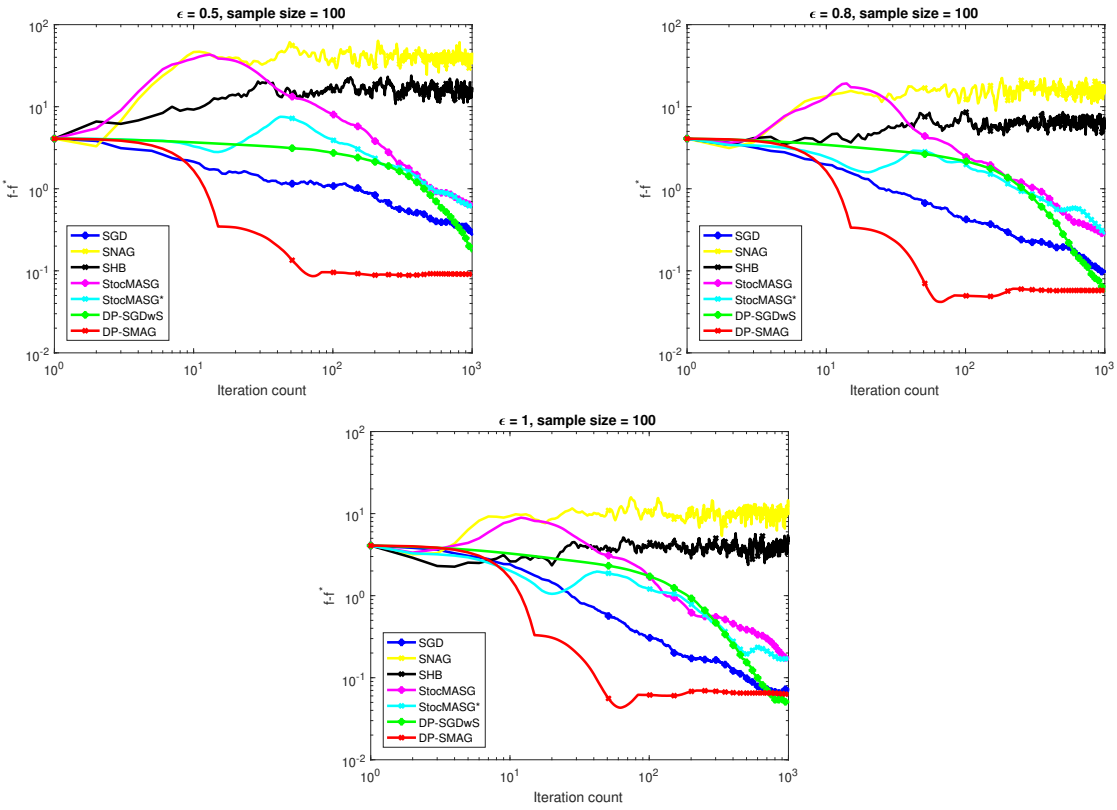


Figure 4.18: Comparison of Stochastic Algorithms For Sample Size = 100

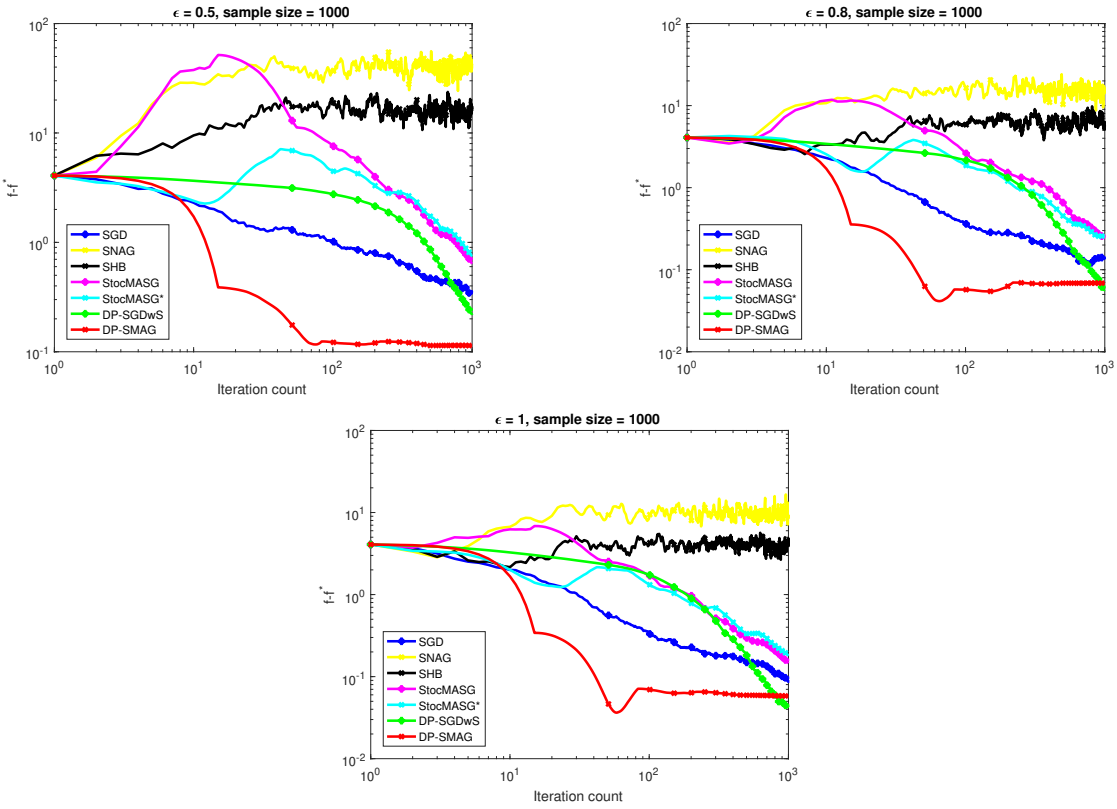


Figure 4.19: Comparison of Stochastic Algorithms For Sample Size = 1000

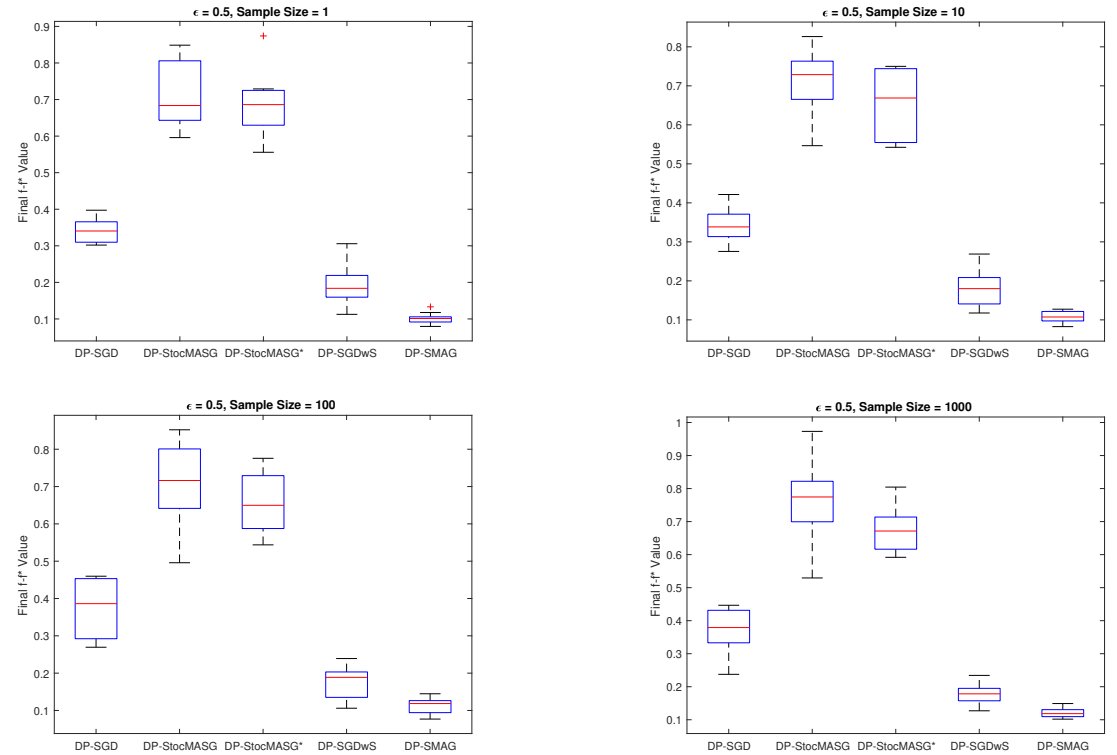


Figure 4.20: Comparison of Stochastic Algorithms For Different Datasets

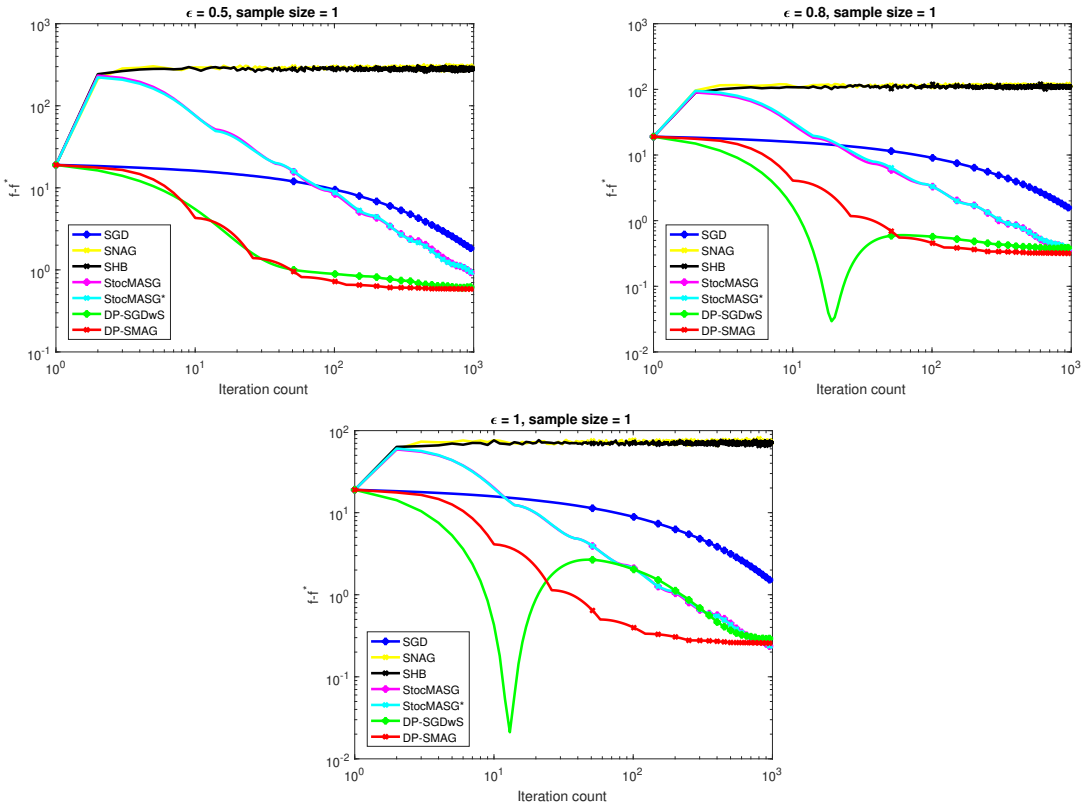


Figure 4.21: Comparison of Stochastic Algorithms For Sample Size = 1 on MNIST

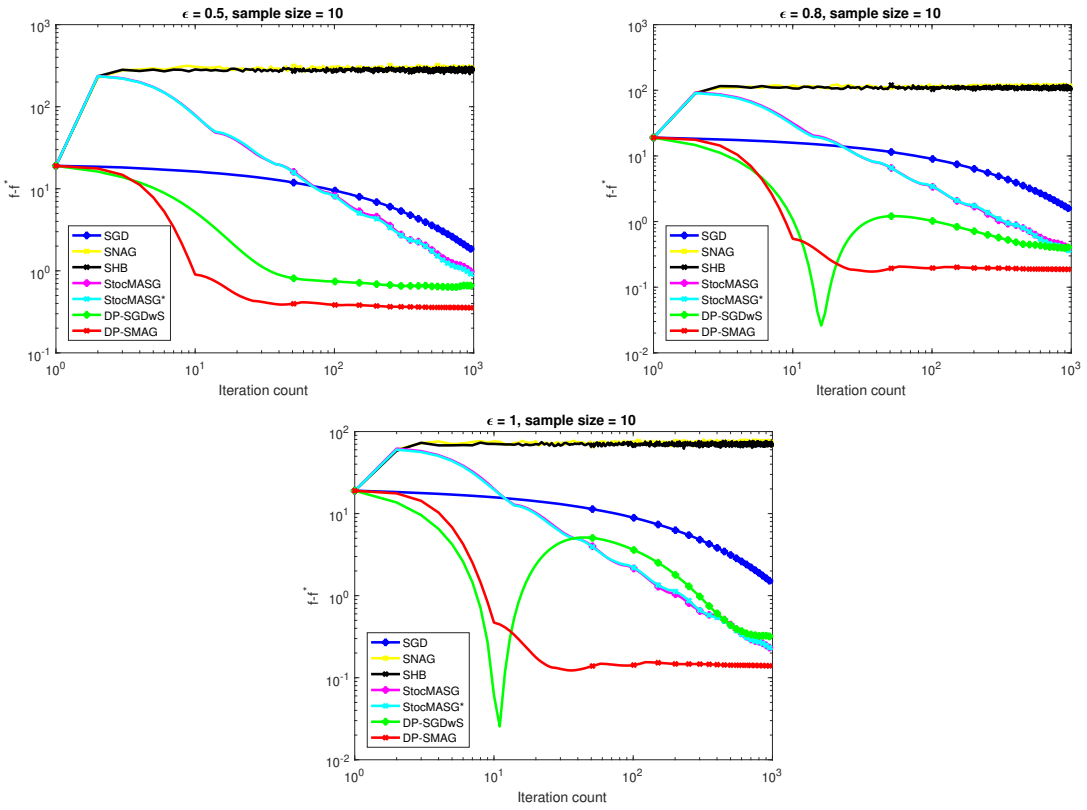


Figure 4.22: Comparison of Stochastic Algorithms For Sample Size = 10 on MNIST

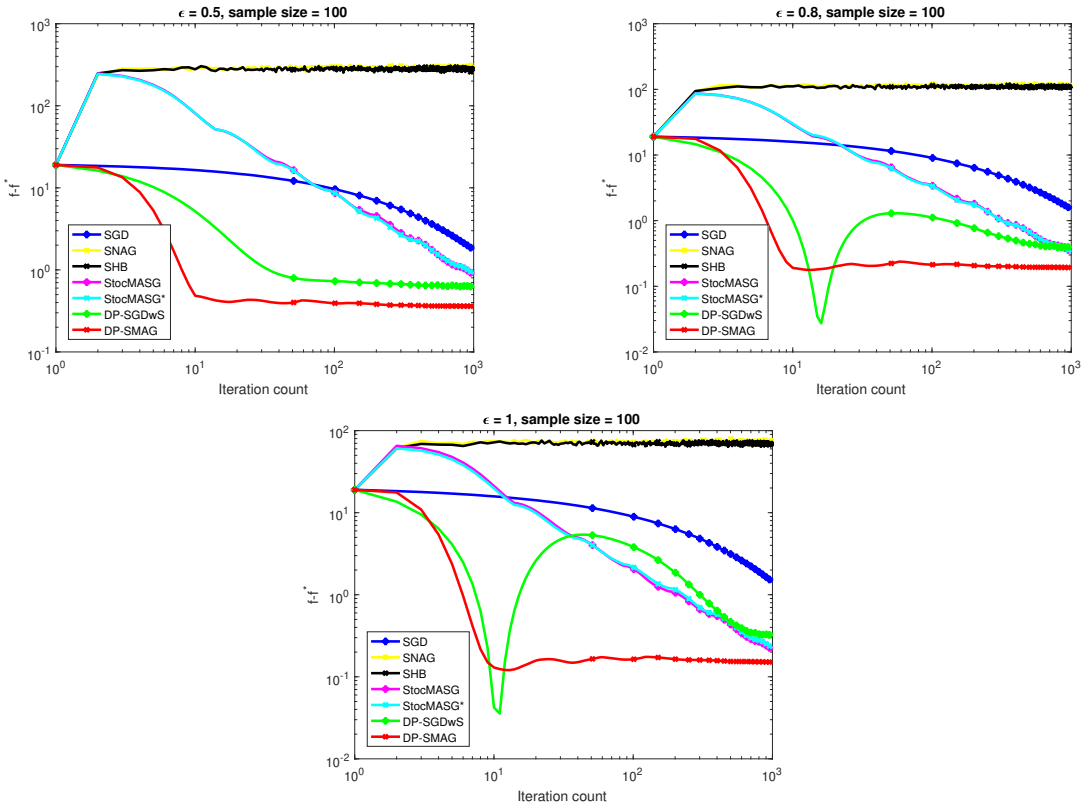


Figure 4.23: Comparison of Stochastic Algorithms For Sample Size = 100 on MNIST

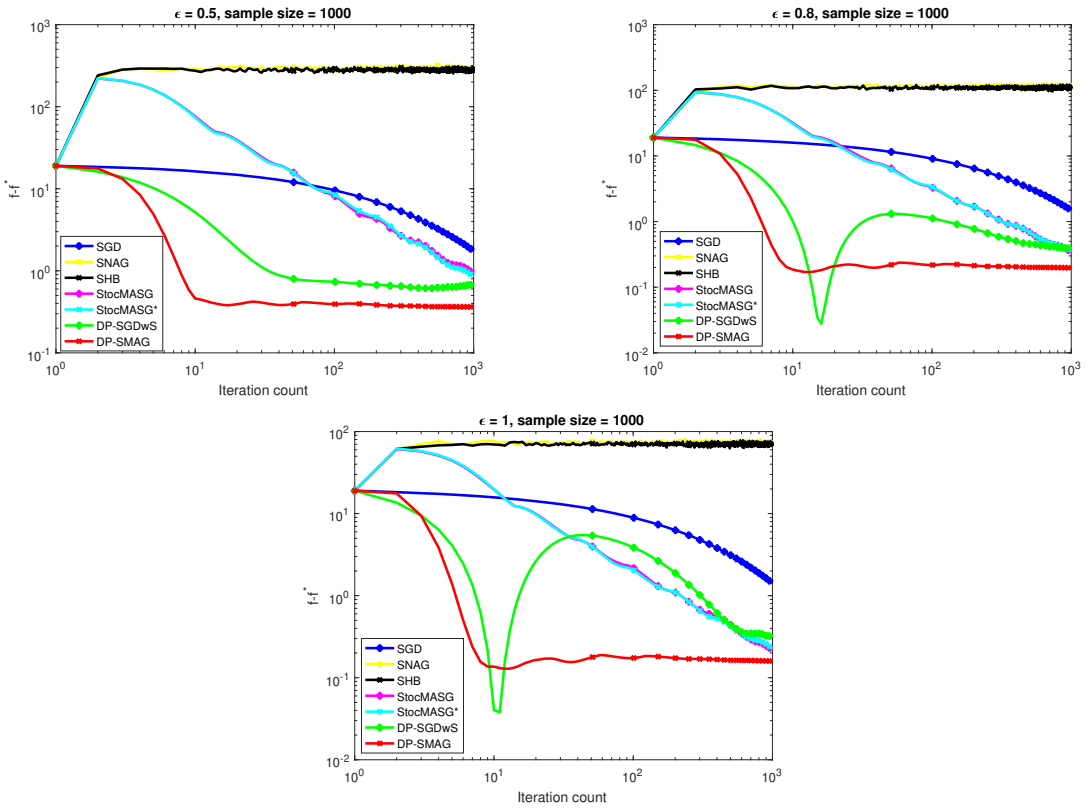


Figure 4.24: Comparison of Stochastic Algorithms For Sample Size = 1000 on MNIST

Appendix A

Omitted Proofs and Results

A.1 Proof of Theorem 4.1.4

This result improves Lemma 34 in [86]. The only difference is that we obtain a tighter privacy level by arranging their proof. We already have the result from this study with subsampling rate $\gamma = \frac{m}{n}$, the new privacy is changed from ϵ to

$$\epsilon' = \log(1 + \kappa[e^{\epsilon} - 1]) - \log(1 + \gamma[e^{-\epsilon} - 1])$$

For $x > 0$, $\log(1 + x) \leq x$ and for $-1 < y < 0$ we have $\log(1 + y) > \frac{y}{1+y}$. Using this,

$$\begin{aligned}
\epsilon' &\leq \gamma(e^{\epsilon} - 1) - \frac{\gamma[e^{-\epsilon} - 1]}{1 + \gamma[e^{-\epsilon} - 1]} \\
&= \gamma[e^{\epsilon} - 1] - \frac{\gamma[1 - e^{\epsilon}]}{e^{\epsilon}(1 + \gamma[e^{-\epsilon} - 1])} \\
&= \gamma[e^{\epsilon} - 1] + \frac{\gamma[1 - e^{\epsilon}]}{e^{\epsilon}(1 + \gamma[e^{-\epsilon} - 1])} \\
&= \gamma[e^{\epsilon} - 1]\left(1 + \frac{1}{e^{\epsilon}(1 + \gamma[e^{-\epsilon} - 1])}\right) \\
&= \gamma[e^{\epsilon} - 1]\left(1 + \frac{1}{e^{\epsilon} + \gamma[1 - e^{\epsilon}]}\right)
\end{aligned}$$

Since

$$e^{\epsilon} + \gamma[1 - e^{\epsilon}] = e^{\epsilon}(1 - \gamma) + \gamma \geq 1,$$

we conclude that

$$\epsilon' \leq 2\gamma[e^{\epsilon} - 1]$$

where $\gamma = \frac{m}{n}$.

A.2 Proof of Proposition 4.3.2

Let us fix $x_1 = \ldots = x_t = x$. We shall use the following identity

$$\mathrm{Var}\left(\widetilde{\nabla f}^{(t)}_{B_{0:t}}(\underbrace{(x,\ldots,x)}_{t+1 \text{ times}}; y_{1:n})\right) = \mathrm{tr}\left\{\mathrm{Cov}\left(\widetilde{\nabla f}^{(t)}_{B_{0:t}}(x_{0:t}; y_{1:n})\right)\right\}. \tag{A.1}$$

Recall that

$$\begin{aligned}\widetilde{\nabla f}^{(t)}_{B_{0:t}}(x_{0:t}; y_{1:n}) &= (1-\beta)\widetilde{\nabla f}^{(t-1)}_{B_{0:t-1}}(x_{0:t-1}; y_{1:n}) + \beta \nabla f_{B_t}(x_t; y_{1:n}) + \eta_t \\ &= (1-\beta)\widetilde{\nabla f}^{(t-1)}_{B_{0:t-1}}(x_{0:t-1}; y_{1:n}) + \beta\left(\nabla f_{B_t}(x_t; y_{1:n}) + \frac{\eta_t}{\beta}\right),\end{aligned}$$

where η_t is Laplace noise. Expanding the right hand side until the first iteration, we obtain

$$\widetilde{\nabla f}^{(t)}_{B_{0:t}}(x_{0:t}; y_{1:n}) = \sum_{k=0}^{t}(1-\beta)^{t-k}\beta\left(\nabla f_{B_k}(x; y_{1:n}) + \frac{\eta_k}{\beta}\right).$$

Taking the expectation leads to

$$\mathbb{E}\left[\widetilde{\nabla f}^{(t)}_{B_{0:t}}(x_{0:t}; y_{1:n})\right] = \left(1-(1-\beta)^{t+1}\right)\frac{1}{n}\sum_{i=1}^{n}\nabla f(x, y_i). \tag{A.2}$$

Moreover, we can simply derive

$$\begin{aligned}\mathrm{Cov}\left(\widetilde{\nabla f}^{(t)}_{B_{0:t}}(x_{0:t}; y_{1:n})\right) &= \mathrm{Cov}\left(\sum_{k=0}^{t}(1-\beta)^{t-k}\beta\left(\nabla f_{B_k}(x; y_{1:n}) + \frac{\eta_k}{\beta}\right)\right) \\ &= \sum_{k=0}^{t}(1-\beta)^{2(t-k)}\beta^2\mathrm{Cov}\left(\nabla f_{B_k}(x; y_{1:n}) + \frac{\eta_k}{\beta}\right) \\ &\leq \beta^2\sum_{k=0}^{\infty}(1-\beta)^{2k}\mathrm{Cov}\left(\nabla f_{B_k}(x; y_{1:n}) + \frac{\eta_k}{\beta}\right) \\ &= \beta^2\frac{1}{1-(1-\beta)^2}\mathrm{Cov}\left(\nabla f_{B_0}(x; y_{1:n}) + \frac{\eta_0}{\beta}\right) \\ &= \frac{\beta}{2-\beta}\mathrm{Cov}\left(\nabla f_{B_0}(x; y_{1:n}) + \frac{\eta_0}{\beta}\right).\end{aligned} \tag{A.3}$$

From here on we drop the subscript from η_0 and B_0 to simplify the notation. Since $\eta \sim \mathrm{Lap}(\sigma)$, we have

$$\mathrm{Cov}\left(\nabla f_B(x; y_{1:n}) + \frac{\eta}{\beta}\right) = \mathrm{Cov}\left(\nabla f_B(x; y_{1:n})\right) + 2\frac{\sigma^2}{\beta^2}I_d. \tag{A.4}$$

Let $R = [r_{i,j}] = \mathrm{Cov}\left(\nabla f_B(x; y_{1:n})\right)$. When the indices in B are sampled without replacement, then the diagonal terms in R, can be written as

$$r_{k,k} = \sigma^2_{y,k}m\left(1 - \frac{m-1}{n-1}\right), \quad k = 1, \ldots, d, \tag{A.5}$$

where $\sigma^2_{y,k}$ is the population variance given by

$$\sigma^2_{y,k} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial f(x;y_i)}{\partial x(k)} - \frac{1}{n}\sum_{j=1}^{n}\frac{\partial f(x;y_j)}{\partial x(k)}\right)^2. \tag{A.6}$$

Given $x \in x$, let

$$S_1^{(k)}(x) = \sup_{y,y'}\left|\frac{\partial f(x;y)}{\partial x(k)} - \frac{\partial f(x;y')}{\partial x(k)}\right|, \quad k = 1,\ldots,d$$

be the L_1-sensitivity of the gradient over the dimension k. Since for each x we have $\sup_{y,y'}\left|\frac{\partial f(x;y_i)}{\partial x(k)} - \frac{\partial f(x;y_j)}{\partial x(k)}\right| \leq S_1^{(k)}(x)$, the population variance in (A.6) can be bounded as

$$\sigma^2_{y,k} \leq \frac{S_1^{(k)}(x)^2}{4}. \tag{A.7}$$

Combining (A.3), (A.4), (A.5), and (A.7), and using the relation

$$S_2(x)^2 = S_1^{(1)}(x) + \ldots + S_1^{(d)}(x),$$

we bound the trace of the covariance as

$$\mathrm{tr}\left\{\mathrm{Cov}\left(\widetilde{\nabla f}^{(t)}_{B_{0:t}}(x_{0:t};y_{1:n})\right)\right\} \leq \left[m\frac{S_2(x)^2}{4}\left(1 - \frac{m-1}{n-1}\right) + 2d\frac{\sigma_t^2}{\beta^2}\right]\frac{\beta}{2-\beta}. \tag{A.8}$$

Bibliography

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[2] N. S. Aybat, A. Fallah, M. Gürbüzbalaban, and A. Ozdaglar. Robust Accelerated Gradient Methods for Smooth Strongly Convex Functions. *arXiv preprint arXiv:1805.10579*, 2018.

[3] N. S. Aybat, A. Fallah, M. Gürbüzbalaban, and A. Ozdaglar. A Universally Optimal Multistage Accelerated Stochastic Gradient Method. *arXiv preprint arXiv:1901.08022*, 2019.

[4] R. Bassily, A. Smith, and A. Thakurta. Differentially Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. *arXiv preprint arXiv:1405.7085*, 2014.

[5] A. Beck and M. Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[6] A. Beimel, S. P. Kasiviswanathan, and K. Nissim. Bounds on the Sample Complexity for Private Learning and Private Data Release. In *Theory of Cryptography Conference*, pages 437–454. Springer, 2010.

[7] A. S. Berahas, M. Jahani, and M. Takáč. Quasi-Newton Methods for Deep Learning: Forget the Past, Just Sample. *arXiv preprint arXiv:1901.09997*, 2019.

[8] A. S. Berahas, J. Nocedal, and M. Takáč. A Multi-Batch L-BFGS Method for Machine Learning. In *Advances in Neural Information Processing Systems 29:*

Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pages 1055–1063, 2016.

[9] D. P. Bertsekas. Incremental Least Squares Methods and the Extended Kalman Filter. *SIAM Journal on Optimization*, 6(3):807–822, 1996.

[10] D. P. Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010:1–38, 2011.

[11] D. Blatt, A. O. Hero, and H. Gauchman. A Convergent Incremental Gradient Method with a Constant Step Size. *SIAM Journal on Optimization*, 18(1):29–51, 2007.

[12] L. Bottou. Online Learning and Stochastic Approximations. *On-line learning in neural networks*, 17(9):142, 1998.

[13] M. Bun and T. Steinke. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[14] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A Stochastic Quasi-Newton Method for Large-Scale Optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[15] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1-3):129–156, 1994.

[16] A. Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[17] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, pages 289–296, 2009.

[18] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially Private Empirical Risk Minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[19] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari. *Nonnegative Matrix and Tensor Factorization.* Wiley, 2009.

[20] S. Cyrus, B. Hu, B. Van Scoy, and L. Lessard. A Robust Accelerated Optimization Algorithm for Strongly Convex Functions. In *2018 Annual American Control Conference (ACC)*, pages 1376–1381. IEEE, 2018.

[21] D. Desfontaines and B. Pejó. SoK: Differential Privacies. *arXiv preprint arXiv:1906.01337*, 2019.

[22] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local Privacy and Statistical Minimax Rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 429–438. IEEE, 2013.

[23] C. Dwork. Differential Privacy. *Springer-Verlag*, 33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006), 2006.

[24] C. Dwork. Differential Privacy: A Survey of Results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[25] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. 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.

[26] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[27] C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[28] C. Dwork and G. N. Rothblum. Concentrated Differential Privacy. *arXiv preprint arXiv:1603.01887*, 2016.

[29] C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and Differential Privacy. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 51–60. IEEE, 2010.

[30] M. Fazlyab, A. Ribeiro, M. Morari, and V. M. Preciado. Analysis of Optimization Algorithms via Integral Quadratic Constraints: Nonstrongly Convex Problems. *SIAM Journal on Optimization*, 28(3):2654–2689, 2018.

[31] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[32] S. Gadat, F. Panloup, and S. Saadane. Stochastic Heavy Ball. *Electronic Journal of Statistics*, 12(1):461–529, 2018.

[33] A. H. Gebremedhin, D. Nguyen, Md. M. A. Patwary, and A. Pothen. ColPack: Software for Graph Coloring and Related Problems in Scientific Computing. *ACM Trans. Math. Softw.*, 40(1):1:1–1:31, October 2013.

[34] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-Scale Matrix Factorization with Distributed Stochastic Gradient Descent. In *ACM SIGKDD*, 2011.

[35] E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson. Global convergence of the Heavy-ball method for convex optimization. In *2015 European Control Conference (ECC)*, pages 310–315. IEEE, 2015.

[36] R. Gower, F. Hanzely, P. Richtárik, and S. U. Stich. Accelerated Stochastic Matrix Inversion: Ggeneral Theory and Speeding up BFGS Rules for Faster Second-Order Optimization. In *Advances in Neural Information Processing Systems*, pages 1619–1629, 2018.

[37] R. M. Gower, D. Goldfarb, and P. Richtárik. Stochastic Block BFGS: Squeezing More Curvature out of Data. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1869–1878, New York, New York, USA, 20–22 Jun 2016. PMLR.

[38] M. Gürbüzbalaban, A. Ozdaglar, and P. Parrilo. A globally convergent incremental Newton method. *Mathematical Programming*, 151(1):283–313, 2015.

[39] F. M. Harper and J. A. Konstan. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.

[40] B. Hu and L. Lessard. Dissipativity Theory for Nesterov's Accelerated Method. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1549–1557. JMLR. org, 2017.

[41] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. Accelerating Stochastic Gradient Descent for Least Squares Regression. *arXiv preprint arXiv:1704.08227*, 2017.

[42] A. Jalilzadeh, A. Nedić, U. V. Shanbhag, and F. Yousefian. A Variable Sample-size Stochastic Quasi-Newton Method for Smooth and Nonsmooth Stochastic Convex Optimization. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4097–4102. IEEE, 2018.

[43] P. Kairouz, S. Oh, and P. Viswanath. The Composition Theorem for Differential Privacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049, 2017.

[44] Y. Kang, Y. Liu, and W. Wang. Weighted Distributed Differential Privacy ERM: Convex and Non-convex. *arXiv preprint arXiv:1910.10308*, 2019.

[45] K. Kaya, F. Öztoprak, Ş. İ. Birbil, A. T. Cemgil, U. Şimşekli, N. Kuru, H. Koptagel, and M. K. Öztürk. A framework for parallel second order incremental optimization algorithms for solving partially separable problems. *Computational Optimization and Applications*, 72(3):675–705, 2019.

[46] R. Kidambi, P. Netrapalli, P. Jain, and S. Kakade. On the insufficiency of existing momentum schemes for stochastic optimization. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–9. IEEE, 2018.

[47] D. Kifer, A. Smith, and A. Thakurta. Private Convex Empirical Risk Minimization and High-dimensional Regression. In *Conference on Learning Theory*, pages 25–1, 2012.

[48] L. Lessard, B. Recht, and A. Packard. Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[49] N. Li, W. Qardaji, and D. Su. On Sampling, Anonymization, and Differential Privacy or, k-Anonymization Meets Differential Privacy. In *Proceedings of the 7th*

ACM Symposium on Information, Computer and Communications Security, pages 32–33. ACM, 2012.

[50] Q. Lin, Z. Lu, and L. Xiao. An Accelerated Randomized Proximal Coordinate Gradient Method and Its Application to Regularized Empirical Risk Minimization. *SIAM Journal on Optimization*, 25(4):2244–2273, 2015.

[51] N. Loizou and P. Richtárik. Linearly convergent stochastic heavy ball method for minimizing generalization error. *arXiv preprint arXiv:1710.10737*, 2017.

[52] O. L. Mangasarian and M. V. Solodov. Serial and parallel backpropagation convergence via nonmonotone perturbed minimization. *Optimization Methods and Software*, 4:103–116, 1994.

[53] F. McSherry and K. Talwar. Mechanism Design via Differential Privacy. In *FOCS*, volume 7, pages 94–103, 2007.

[54] I. Mironov. Rényi Differential Privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[55] A. Mokhtari, M. Eisen, and A. Ribeiro. IQN: An Incremental Quasi-Newton Method with Local Superlinear Convergence Rate. *arXiv preprint arXiv:1702.00709*, 2017.

[56] P. Moritz, R. Nishihara, and M. I. Jordan. A linearly-convergent stochastic l-bfgs algorithm. In *Artificial Intelligence and Statistics*, pages 249–258, 2016.

[57] I. Necoara and V. Nedelcu. Rate Analysis of Inexact Dual First-Order Methods Application to Dual Decomposition. *IEEE Transactions on Automatic Control*, 59(5):1232–1243, 2013.

[58] A. S. Nemirovsky and D. B. Yudin. Problem Complexity and Method Efficiency in Optimization. 1983.

[59] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, 1993.

[60] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer, 2004.

[61] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[62] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth Sensitivity and Sampling in Private Data Analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2007.

[63] S. Osher, B. Wang, P. Yin, X. Luo, F. Barekat, M. Pham, and A. Lin. Laplacian Smoothing Gradient Descent. *arXiv preprint arXiv:1806.06317*, 2018.

[64] M. Park, J. Foulds, K. Chaudhuri, and M. Welling. Variational Bayes in Private Settings (VIPS). *arXiv preprint arXiv:1611.00340*, 2016.

[65] V. Pichapati, A. T. Suresh, F. X. Yu, S. J Reddi, and S. Kumar. AdaClip: Adaptive Clipping for Private SGD. *arXiv preprint arXiv:1908.07643*, 2019.

[66] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[67] A. Ramezani-Kebrya, A. Khisti, and B. Liang. On the Stability and Convergence of Stochastic Gradient Descent with Momentum. *arXiv preprint arXiv:1809.04564*, 2018.

[68] N. L. Roux, M. Schmidt, and F. R. Bach. A Stochastic Gradient Method with an Exponential Convergence Rate for Finite Training Sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

[69] B. I. P. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft. Learning in a Large Function Space: Privacy-Preserving Mechanisms for SVM Learning. *arXiv preprint arXiv:0911.5708*, 2009.

[70] N. N. Schraudolph, J. Yu, and S. Günter. A Stochastic Quasi-Newton Method for Online Convex Optimization. In *Proceedings of the 11th International Conference Artificial Intelligence and Statistics (AISTATS)*, pages 433–440, 2007.

[71] O. Shamir, N. Srebro, and T. Zhang. Communication Efficient Distributed Optimization using an Approximate Newton-type Method. In *International Conference on Machine Learning (ICML)*, 2014.

[72] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321. ACM, 2015.

[73] A. P. Singh and G. J. Gordon. A Unified View of Matrix Factorization Models. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Part II*, pages 358–373. Springer, 2008.

[74] J. Sohl-Dickstein, B. Poole, and S. Ganguli. Fast large-scale optimization by unifying stochastic gradient and quasi-Newton methods. In *Proceedings of the 31th International Conference on Machine Learning (ICML)*, pages 604–612, 2014.

[75] M. V. Solodov. Incremental Gradient Algorithms with Stepsizes Bounded Away from Zero. *Computational Optimization and Applications*, 11(1):23–35, 1998.

[76] S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 245–248. IEEE, 2013.

[77] S. Song, K. Chaudhuri, and A. D. Sarwate. Learning from Data with Heterogeneous Noise Using SGD. In *Artificial Intelligence and Statistics*, pages 894–902, 2015.

[78] T. Sun, P. Yin, D. Li, C. Huang, L. Guan, and H. Jiang. Non-ergodic Convergence Analysis of Heavy-Ball Algorithms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5033–5040, 2019.

[79] K. Talwar, A. Thakurta, and L. Zhang. Private Empirical Risk Minimization Beyond the Worst Case: The Effect of the Constraint Set Geometry. *arXiv preprint arXiv:1411.5417*, 2014.

[80] P. Tseng. An Incremental Gradient (-Projection) Method with Momentum Term and Adaptive Stepsize Rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

[81] B. Van Scoy, R. A. Freeman, and K. M. Lynch. The Fastest Known Globally Convergent First-Order Method for Minimizing Strongly Convex Functions. *IEEE Control Systems Letters*, 2(1):49–54, 2017.

[82] B. Wang, Q. Gu, M. Boedihardjo, F. Barekat, and S. J. Osher. DP-LSSGD: A Stochastic Optimization Method to Lift the Utility in Privacy-Preserving ERM. *arXiv preprint arXiv:1906.12056*, 2019.

[83] D. Wang, M. Ye, and J. Xu. Differentially Private Empirical Risk Minimization Revisited: Faster and More General. In *Advances in Neural Information Processing Systems*, pages 2722–2731, 2017.

[84] L. Wang, B. Jayaraman, D. Evans, and Q. Gu. Efficient Privacy-Preserving Nonconvex Optimization. *arXiv preprint arXiv:1910.13659*, 2019.

[85] X. Wang, S. Ma, D. Goldfarb, and W. Liu. Stochastic Quasi-Newton Methods for Nonconvex Stochastic Optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

[86] Y. Wang, J. Lei, and S. E. Fienberg. Learning with Differential Privacy: Stability, Learnability and the Sufficiency and Necessity of ERM Principle. *Journal of Machine Learning Research*, 17, 2016.

[87] A. Wills, C. Jidling, and T. Schon. A fast quasi-newton-type method for large-scale stochastic optimisation. *arXiv preprint arXiv:1810.01269*, 2018.

[88] T. Yang, Q. Lin, and Z. Li. Unified Convergence Analysis of Stochastic Momentum Methods for Convex and Non-convex Optimization. *arXiv preprint arXiv:1604.03257*, 2016.

[89] F. Yousefian, A. Nedić, and U. V. Shanbhag. Stochastic quasi-Newton methods for non-strongly convex problems: convergence and rate analysis. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 4496–4503. IEEE, 2016.

[90] L. Yu, L. Liu, C. Pu, M. E. Gürsoy, and S. Truex. Differentially Private Model Publishing for Deep Learning. *arXiv preprint arXiv:1904.02200*, 2019.

[91] S. K. Zavriev and F. V. Kostyuk. Heavy-ball method in nonconvex optimization problems. *Computational Mathematics and Modeling*, 4(4):336–341, 1993.

[92] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Functional Mechanism: Regression Analysis under Differential Privacy. *Proceedings of the VLDB Endowment*, 5(11):1364–1375, 2012.

[93] J. Zhang, K. Zheng, W. Mou, and L. Wang. Efficient Private ERM for Smooth Objectives. *arXiv preprint arXiv:1703.09947*, 2017.