THE UNIVERSITY OF CHICAGO

HOMOGENEOUS AND COUPLED DISTRIBUTED LEARNING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
JIALEI WANG

CHICAGO, ILLINOIS
JUNE 2018

Dedicated to my parents and my wife

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First I would like to thank my advisors Nati Srebro and Mladen Kolar for their encouragement and guidance during the past five years of my PhD adventure, I was extremely fortunate to be advised by them. Nati and Mladen guided my every detail on research at the early stage: from asking research questions to formalizing problems, from scientific writing to presenting ideas. I am grateful to them for giving me freedom on choosing the research topic at later stage. I also benefit a lot from Nati's learning theory and convex optimization courses, which greatly influenced my thinking on statistics and computation. I appreciate Janos Simon and Mike Franklin for standing in my thesis committee.

During my PhD study, I have spent several wonderful research internships in industry labs. I would like to thank Ryohei Fujimaki at NEC Labs, Peder Olsen, Andrew Conn and Aurelie Lozano at IBM Research, Lin Xiao at Microsoft Research, Brendan McMahan at Google Research for hosting me in summers. Special thanks to Tong Zhang at Tencent AI Lab, I have learned a lot from Tong's sharp theoretical insight. I would also like to thank my collaborators Samory Kpotufe, Shubhendu Trived and Greg Shakhnarovich, James Evans, Dan Garber, Jason Lee, Mehrdad Mahdavi, Yining Wang, Chao Gao, Shun Zheng, Jianqiao Wangni and Ji Liu who shaped my research. Special thanks to Weiran Wang, for the effective research collaboration through numerous discussions.

I sincerely thank my fellow students and friends at the university for their help and suggestions on numerous things. Qinqing Zheng and Jiajun Shen helped me a lot during the first year of my PhD study. I would like to thank Zhixuan Zhou and Liwen Zhang for being amazing roommates and made my life much easier at UChicago. Special thanks to Yuanwei Fang, for being my best friend at Chicago, for making my PhD life so much fun.

Finally, my deepest gratitude goes to my parents and my wife for their love, support and encouragement. I would never be here without their constant cultivation and belief, this thesis would also not have been possible with out them.

ABSTRACT

This thesis explores the theoretical foundations of distributed machine learning with practical considerations. Distributed learning systems are able to handle data sets that cannot be processed on a single machine, and utilize parallel computing resources to speed up the learning process. However, it also brings unique challenges due to the characteristics of modern data sets and distributed computing infrastructure: on one hand, machines are required to being able to extract meaningful parsimonious structures from the large-scale, high-dimensional data; on the other hand, the heterogeneity in distributed data sets enforce us to consider flexible models that are adaptive to each local machine; last but not least, the overall effectiveness in distributed learning systems depends on the efficiency of learning algorithms on multiple resources: computing, communication, sample and memory, and we need to design algorithms that balance multiple efficiency constraints.

In this thesis, we considered distributed machine learning under both homogeneous and coupled setting. In the homogeneous setting, each machine has access to an independent local data set drawn from the same source distribution, while in the coupled scenario, the local data sets might drawn from different distributions and the goal is to extract the common structure through distributed learning. In this thesis, we study the trade-offs between sample complexity, computational cost, communication and memory efficiency in both settings; we propose novel methods that effectively leverage the similarity/relatedness structure between machines for several distributed learning problems, with improved theoretical guarantees. We also examine the practical performance of the proposed approaches with existing methods via numerical experiments.

CHAPTER 1
THESIS OVERVIEW

The advent of big data enables machine learns to perform complex tasks such as visual understanding and machine translation, extract knowledge from scientific and social-economical domains. Such a key empirical success can be attributed to several key factors: the modern data acquisition technologies helped scientists and engineers collected large-scale high quality data sets; the development of learning algorithms that enables machines to recognize hidden patterns from noisy observations; the rapid growth of computing power allows us to perform large-scale training of complex models. For example, multiple computing machines have been employed to train high-dimensional sparse linear models for click-through rate prediction problems in advertisement display [McMahan et al., 2013, Li et al., 2014], GPU clusters are used to train structured neural networks with huge number of parameters in computer vision and natural language processing [Krizhevsky et al., 2012, Sutskever et al., 2014]. Therefore, distributed machine learning have been an emerging technology for many applications with large-scale data sets.

Distributed learning can leverage distributed computing resources, handling data sets that cannot be processed on a single machine, thus speed up the learning process. Moreover, when data are naturally collected from decentralized resources such as mobile phones and sensor networks, distributed learning enables us to construct predictive models without communicating the local data thus ensuring better privacy. For example, federated learning have been deployed for modeling users' querying behavior, thus create better search experience on the mobile phones [Konečnỳ et al., 2016, McMahan et al., 2017].

1.1 Challenges of distributed machine learning

Although distributed machine learning have been widely applied in industry, it also brings unique challenges in developing distributed learning algorithms. In the following, we summarize the following critical challenges which we will address in this thesis.

1.1.1 Large-scale, high-dimensional, heterogeneous data

Nowadays, we often encountered datasets that have contains a lot of instances. For example, industry level computational ads displaying models are trained on billions of examples [McMahan et al., 2013]; intelligent recommendation systems are built upon multiple millions of users and millions of items [Davidson et al., 2010, Smith and Linden, 2017]; state-of-the-art image classification models are learned using millions of images [Russakovsky et al., 2015]. To train a prediction model from such a huge database, traditional optimization approaches such as gradient descent are often infeasible since calculating the full gradient require making a pass over the whole data which is very expensive. Thus the optimization methods of choice for large-scale machine learning problems are often some randomized optimization methods such as stochastic gradient descent and its variants. Stochastic optimization methods process a small fraction of the datasets at every iteration, and thus have much cheaper cost per iteration. Unfortunately, stochastic gradient descent is sequential in nature, thus it is important to design new optimization methods that can be implemented in parallel computing platforms. Moreover, when we focus on empirical risk minimization of finite-sum objectives, stochastic optimization with variance reduction techniques [Johnson and Zhang, 2013] have been recently proposed as the state-of-the-art solvers. Designing efficient distributed optimization algorithms for solving empirical risk minimization problems has also becomes an important research problem.

Besides the large sample size challenge (the "big n problem"), another distinguishing feature of modern datasets is its high-dimensionality (the "big p problem"). For many complex modeling problems, the number of features of variables used in the prediction problem is large, often of the similar order or even larger than the sample size. For example, for click-through rate prediction problem, a lot of descriptive features have been collected, including bag-of-words features, making the problem dimension to be also at billions scale [Li et al., 2014]. It is well known that in high-dimensions, the rate of convergence in statistical estimation is very slow (both in parametric and non-parametric models), and such a difficulty in performing data analysis in high-dimensions is often referred to the "curse of dimensionality" phenomena. To tackle this challenge, a key technique proposed is to extract low-dimensional structures from high-dimensional data [Bühlmann and Van De Geer, 2011, Hastie et al., 2015], and it has been a major research theme in statistics, machine learning and signal processing. Thus it is natural to ask how to perform high-dimensional statistical learning tasks in a distributed environment.

Another key property of the distributed data is the potential heterogeneity. The success of standard supervised machine learning paradigm relies heavily on the i.i.d. assumption on the training data as well as the future testing data. However, when the data are collected in a distributed fashion, it is sometimes no longer reasonable to assume the data from difference machines came from the same distribution. Taking the on-device intelligence problem as an example, user data collected from different mobile phones are unlikely to have the same distribution. Rather, it is more reasonable to argue each user holds a separate distribution, according to which the user behavior data are generated. But still, different users might share some common characteristics in the data generating process, and it is important to take advantages over the shared structure in the learning process. Thus, the heterogeneity in distributed data brings interesting research question in designing coupled distributed learning algorithms.

1.1.2 Multiple recourse constraints

Typically, a learning algorithm is evaluated based on its statistical accuracy and computational efficiency. The statistical accuracy measures the how good prediction the model make on future unseen data, or how close is the estimated model to the ground truth. The computational efficiency was traditionally measured by the wall clock time run on a single processor. In the distributed computing scenario, there is another dimension called communication efficiency also affect significantly on the total wall clock time. The communication cost measures the overhead of exchanging information across machines. In a variety of distributed systems, the communication cost of exchanging a single message is much more expensive than the floating point operations [Martin et al., 1997]. As a consequence, the communication efficiency has been a major bottleneck for many distributed learning systems. To sum up, we face multiple recourse constraints when developing distributed learning algorithms, and these efficiency requirements are often conflict with each other and there might be some inherent trade-off between them:

- **Sample efficiency** We would like the distributed algorithms to utilize the available data efficiently, and the goal is be as statistically accurate as the centralized solution, which put all data together and solve the problem on a single machine.

- **Computation efficiency** We would like to use multiple available machines to share the computation workload in the training process. Ideally we would like to have a linear reduction in computation compared with best possible single machine solutions.

- **Communication efficiency** Communication cost consists of latency cost and bandwidth cost, we could want to keep the total communication cost of the distributed learning algorithm as low as possible.

- **Memory efficiency and other constraints** We might have some constraints in some situations. For example, like stochastic gradient descent on single machine, we would like to load the data only once to keep memory efficiency; sometimes the machines would like not to disclosing too much information about their local data, which is often formulated as guarantees on differential privacy.

With such recourse constraints on multiple dimensions, it becomes challenging in developing distributed learning algorithms that are good at above all aspects, sometimes natural trade-offs exists between one kind of recourse (e.g. communication) to another (computation).

1.2 Traditional approaches

A common approach to distributed learning is to minimize the empirical risk of data distributed across machines, by viewing the total empirical risk as an average of local per-machine empirical risks:

$$f_i(\mathbf{w}) := \frac{1}{n}\sum_{j}^{n} \ell(\mathbf{w}, \mathbf{z}_{ij}),$$

where j is the index of individual data instance $\mathbf{z}_{ij}$, n is the sample size on i-th machine and $\ell(\cdot,\cdot)$ is the loss function. Empirical risk minimization (ERM) then reduces to the distributed optimization problem

$$\min_{\mathbf{w}} f(\mathbf{w}) := \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{w}) \tag{1.1}$$

where each machine holds the local objective $f_i(\cdot)$. There is indeed much work on **distributed consensus** problems of the form (1.1) [Tsitsiklis et al., 1986, Bertsekas and Tsitsiklis, 1997, Xiao et al., 2007, Nedic et al., 2010, Shi et al., 2014]. The main difficulty of consensus problems is that in general, the local objectives might be arbitrarily different, and as a result, one can obtain strong lower bounds on the amount of communication that must be exchanged in order to reach a joint optimum [Olshevsky and Tsitsiklis, 2011]. In particular, the problem becomes *harder* as more machines are involved, requiring more communication and more computation.

But as we argue here, simply reducing distributed learning to such a deterministic consensus problem overlooks the stochastic nature of the learning problem. As discussed above, guarantees for deterministic consensus necessarily becomes *worse* as more machine are available. But a goal of distributed learning is to *reduce* the cost of learning by having more machines available. Any approach based on generic deterministic consensus analysis is thus futile in harnessing the benefit of distributed learning.

There are actually two issues that are ignored here: first, that our true objective isn't just getting small empirical error, but rather using samples to minimize our true objective which is the expected risk (e.g. generalization error, error on future examples). As we now understand in the serial setting, keeping this true objective in mind changes both our analysis and our choice of methods [Bousquet and Bottou, 2008, Shalev-Shwartz et al., 2011b]. Focusing on the true objective also allows us to understand how, at least in the serial setting, having more data available does not mean runtime must increase (as it would if we focused only on the empirical objective), but actually makes the problem easier and even allows decreasing runtime [Shalev-Shwartz and Srebro, 2008]. Second, the local objectives $f_i(\mathbf{w})$ are *not* arbitrarily different, as in the deterministic consensus setting, as they often reflect data drawn from the same or similar source distributions. Taking both issues into account will allow us to develop and study methods that leverage additional machines in order to make the problem easier, not harder. In this thesis, we ask and pursuit answer for the following key question in distributed learning:

How to leverage the similarity/relatedness structures between machines when designing distributed machine learning algorithms ?

1.3 Overview and organization of main results

In this thesis, we make the following contributions to the field of distributed machine learning:

- **Efficient distributed learning with sparsity.** We propose a novel, efficient approach for distributed sparse learning with observations randomly partitioned across machines. In each round of the proposed method, worker machines compute the gradient of the loss on local data and the master machine solves a shifted ℓ_1 regularized loss minimization problem. After a number of communication rounds that scales only logarithmically with the number of machines, and independent of other parameters of the problem, the proposed approach provably matches the estimation error bound of centralized methods. We present and analyze an approach for distributed stochastic optimization which is statistically optimal and achieves near-linear speedups (up to logarithmic factors). Our approach allows a communication-memory tradeoff, with either logarithmic communication but linear memory, or polynomial communication and a corresponding polynomial reduction in required memory. This communication-memory tradeoff is achieved through minibatch-prox iterations (minibatch passive-aggressive updates), where a subproblem on a minibatch is solved at each iteration. We provide a novel analysis for such a minibatch-prox procedure which achieves the statistical optimal rate regardless of minibatch size and smoothness, thus significantly improving on prior work. This work appeared in [Wang et al., 2017a], and is presented in detail in Chapter 2.

- **Memory and communication efficient distributed stochastic optimization with minibatch-prox.** We present and analyze an approach for distributed stochastic optimization which is statistically optimal and achieves near-linear speedups (up to logarithmic factors). Our approach allows a communication-memory tradeoff, with either logarithmic communication but linear memory, or polynomial communication and a corresponding polynomial reduction in required memory. This communication-memory tradeoff is achieved through minibatch-prox iterations (minibatch passive-aggressive updates), where a subproblem on a minibatch is solved at each iteration.

We provide a novel analysis for such a minibatch-prox procedure which achieves the statistical optimal rate regardless of minibatch size and smoothness, thus significantly improving on prior work. This work appeared in [Wang et al., 2017d], and is presented in detail in Chapter 3.

- **Distributed optimization with sketching.** We study sketching from an optimization point of view. We first show that the iterative Hessian sketch is an optimization process with preconditioning and develop an accelerated version using this insight together with conjugate gradient descent. Next, we establish a primal-dual connection between the Hessian sketch and dual random projection, which allows us to develop an accelerated iterative dual random projection method by applying the preconditioned conjugate gradient descent on the dual problem. Finally, we tackle the problems of large sample size and high-dimensionality in massive data sets by developing the primal-dual sketch. The primal-dual sketch iteratively sketches the primal and dual formulations and requires only a logarithmic number of calls to solvers of small sub-problems to recover the optimum of the original problem up to arbitrary precision. Our iterative sketching techniques can also be applied for solving distributed optimization problems where data are partitioned by samples or features. This work appeared in [Wang et al., 2017b,c], and is presented in detail in Chapter 4.

- **Communication-computation balanced optimization.** We present novel distributed optimization algorithms to solve empirical risk minimization problems, the methods always achieve near-linear computation speedup even for objectives with large condition number. Empirical results are provided to demonstrate the proposed approach achieves a good balance between communication and computation. This work appeared in [Wang et al., 2018], and is presented in detail in Chapter 5.

- **Distributed multi-task learning with shared sparsity.** We consider the problem of distributed multitask learning, where each machine learns a separate, but related, task. Specifically, each machine learns a linear predictor in high-dimensional space, where all tasks share the same small support. We present a communication-efficient estimator based on the debiased lasso and show that it is comparable with the optimal centralized method. This work appeared in [Wang et al., 2016b], and is presented in detail in Chapter 6.

- **Distributed multi-task learning with shared subspace.** We study the problem of distributed multi-task learning with shared representation, where each machine aims to learn a separate, but related, task in an unknown shared low-dimensional subspaces, i.e. when the predictor matrix has low rank. We consider a setting where each task is handled by a different machine, with samples for the task available locally on the machine, and study communication-efficient methods for exploiting the shared structure. This work appeared in [Wang et al., 2016a], and is presented in detail in Chapter 7.

Some concluding remarks and future directions are presented in Chapter 8.

Empirical vs population objectives Distributed learning algorithms typically construct the model through minimizing certain forms of optimization objectives. One form of objectives is the population objectives, which take expectation over individual loss functions induced by each data instance, the population objective is directly connected to *generalization* [Bousquet and Bottou, 2008], i.e. the ability to predicting the future. Another form of objective is the empirical objective, which takes empirical average of the loss functions induced by observed data. With proper forms of regularization, minimizing empirical objective can also lead to generalization [Vapnik, 1995]. Thus distributed learning algorithms can either work on empirical objectives or directly work on population objectives. The main results of Chapter 2, 3, 6 and 7 focus on the population objectives, while the main results of Chapter 4 and 5 focus on empirical objectives.

CHAPTER 2
EFCIENT DISTRIBUTED LEARNING WITH SPARSITY

2.1 Motivation and problem set-up

We consider learning a sparse linear regressor $\mathbf{w}$ minimizing the population objective:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbb{E}_{\mathbf{X},Y\sim\mathcal{D}} \left[\ell(Y, \langle \mathbf{X}, \mathbf{w}\rangle)\right], \tag{2.1}$$

where $(\mathbf{X}, Y) \in \mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^p \times \mathcal{Y}$ are drawn from an unknown distribution $\mathcal{D}$ and $\ell(\cdot,\cdot)$ is a convex loss function, based on N i.i.d. samples $\{\mathbf{x}_i, y_i\}_{i=1}^N$ drawn from $\mathcal{D}$, and when the support $S := \text{support}(\mathbf{w}^*) = \{j \in [p] \mid \mathbf{w}_j^* \neq 0\}$ of $\mathbf{w}^*$ is small, $|S| \leq s$. In a standard single-machine setting, a common empirical approach is to minimize the ℓ_1 regularized empirical loss (see, e.g., (2.2) below). Here we consider a setting where data are distributed across m machines, and, for simplicity, assume[1] that $N = nm$, so that each machine j has access to n i.i.d. observations (from the same source $\mathcal{D}$) $\{\mathbf{x}_{ji}, y_{ji}\}_{i=1}^n$ (equivalently, that $N = nm$ samples are randomly partitioned across machines).

The main contribution of the chapter is a novel algorithm for estimating $\mathbf{w}^*$ in a distributed setting. Our estimator is able to achieve the performance of a centralized procedure that has access to all data, while keeping computation and communication costs low. Compared to the existing one-shot estimation approach [Lee et al., 2017b], our method can achieve the same statistical performance without performing the expensive debiasing step. As the number of communication rounds increases, the estimation accuracy improves until matching the performance of a centralized procedure, which happens after the logarithm of the total number of machines rounds. Furthermore, our results can be achieved under weak assumptions on the data generating procedure.

1. Results in this chapter easily generalize to a setting where each machine has a different number of observations.

Approach	$n \gtrsim ms^2 \log p$		$ms^2 \log p \gtrsim n \gtrsim s^2 \log p$	
	Communication	Computation	Communication	Computation
Centralize	$n \cdot p$	$\mathtt{T}_{\mathtt{lasso}}(mn, p)$	$n \cdot p$	$\mathtt{T}_{\mathtt{lasso}}(mn, p)$
Avg-Debias	p	$p \cdot \mathtt{T}_{\mathtt{lasso}}(n, p)$	$\times$	$\times$
Ours (EDSL)	p	$2 \cdot \mathtt{T}_{\mathtt{lasso}}(n, p)$	$\log m \cdot p$	$\log m \cdot \mathtt{T}_{\mathtt{lasso}}(n, p)$

Table 2.1: Comparison of resources required for matching the centralized error bound of various approaches for high-dimensional distributed sparse linear regression problems, where $\mathtt{T}_{\mathtt{lasso}}(n, p)$ is the runtime for solving a generalized lasso problem of size $n \times p$.

We assume that the communication occurs in rounds. In each round, machines exchange messages with the master machine. Between two rounds, each machine only computes based on its local information, which includes local data and previous messages [Zhang et al., 2013g, Shamir and Srebro, 2014, Arjevani and Shamir, 2015]. In a non-distributed setting, efficient estimation procedures need to balance statistical efficiency with computation efficiency (runtime). In a distributed setting, the situation is more complicated and we need to balance two resources, local runtime and number of rounds of communication, with the statistical error. The local runtime refers to the amount of work each machine needs to do. The number of rounds of communication refers to how often do local machines need to exchange messages with the master machine. We compare our procedure to other algorithm using the aforementioned metrics.

We consider the following two baseline estimators of $\mathbf{w}^*$: the local estimator uses data available only on the master (first) machine and ignores data available on other machines. In particular, it computes

$$\widehat{\mathbf{w}}_{\text{local}} = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_{1i}, \langle \mathbf{x}_{1i}, \mathbf{w} \rangle) + \lambda ||\mathbf{w}||_1 \tag{2.2}$$

using locally available data. The local procedure is efficient in both communication and computation, however, the resulting estimation error is large compared to an estimator that uses all of the available data. The other idealized baseline is the centralized estimator

$$\widehat{\mathbf{w}}_{\text{centralize}} = \arg\min_{\mathbf{w}} \frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} \ell(y_{ji}, \langle \mathbf{x}_{ji}, \mathbf{w} \rangle) + \lambda ||\mathbf{w}||_1.$$

Unfortunately, due to data being huge and communication expensive, we cannot compute the centralized estimator, even though it achieves the optimal statistical error.

In a related setting, Lee et al. [2017b] studied a one-shot approach to learning $\mathbf{w}^*$, called *Avg-Debias*, that is based on averaging the debiased lasso estimators [Zhang and Zhang, 2013]. Under strong assumptions on the data generating procedure, their approach matches the centralized error bound after one round of communication. While an encouraging result, there are limitations to this approach, that we list below.

- The debiasing step in Avg-Debias is computationally heavy as it requires each local machine to estimate a $p \times p$ matrix. For example, Javanmard [2014] (section 5.1) transforms the problem of estimating the debiasing matrix Θ into p generalized lasso problems. This is computationally prohibitive for high-dimensional problems [Zhang and Zhang, 2013, Javanmard and Montanari, 2014]. In comparison, our procedure requires only solving one ℓ_1 penalized objective in each iteration, which has the same time complexity as computing $\widehat{\beta}_{\text{local}}$ in (2.2). See Section 2.2 for details.

- Avg-Debias procedure only matches the statistical error rate of the centralized procedure when the sample size per machine satisfies $n \gtrsim ms^2 \log p$. Our approach improves this sample complexity to $n \gtrsim s^2 \log p$.

- Avg-Debias procedure requires strong conditions on the data generating process. For example, the data matrix is required to satisfy the generalized coherence condition for debiasing to work[2]. As we show here, such a condition is not needed for consistent

2. The generalized coherence states that there exists a matrix Θ, such that $||\widehat{\Sigma}\Theta - I_p||_\infty \lesssim \sqrt{\frac{\log p}{n}}$, where $\widehat{\Sigma}$ is the empirical covariance matrix.

high-dimensional estimation in a distributed setting. Instead, we only require standard restricted eigenvalue condition that are commonly assumed in the high-dimensional estimation literature.

Our method (EDSL) addresses the aforementioned issues of Avg-Debias. Table 2.1 summarizes the resources required for the approaches discussed above to solve the distributed sparse linear regression problems.

Parallel Work In parallel work (publicly announced on arXiv simultaneously with the results in this contribution), Jordan et al. [2018] present a method which is equivalent to the first iteration of our method, and thus achieves the same computational advantage over Avg-Debias as depicted in the left column of Table 2.1 and discussed in the first and third bullet points above. Jordan et al. extend the idea in ways different and orthogonal to this submission, by considering also low-dimensional and Bayesian inference problems. Still, for high-dimensional problems, they only consider a one-shot procedure, and so do not achieve statistical optimality in the way our method does, and do not allow using $n \lesssim ms^2 \log p$ samples per machine (see right half of Table 2.1). The improved one-shot approach is thus a parallel contribution, made concurrently by Jordan et al. and by us, while the multi-step approach and accompanied reduction in required number of samples (discusse in the second bullet point above) and improvement in statistical accuracy is a distinct contribution of this this submission.

Other Related Work A large body of literature exists on distributed optimization for modern massive data sets [Dekel et al., 2012, Duchi et al., 2012, 2014, Zhang et al., 2013g, Zinkevich et al., 2010, Boyd et al., 2011, Balcan et al., 2012, Yang, 2013, Jaggi et al., 2014, Ma et al., 2015, Shamir and Srebro, 2014, Zhang and Xiao, 2015, Lee et al., 2017a, Arjevani and Shamir, 2015]. A popular approach to distributed estimation is averaging estimators formed locally by different machines [Mcdonald et al., 2009, Zinkevich et al., 2010, Zhang et al., 2012, Huang and Huo, 2015]. Divide-and-conquer procedures also found applications in statistical inference [Zhao et al., 2014a, Shang and Cheng, 2017, Lu et al., 2016]. Shamir and Srebro [2014] and Rosenblatt and Nadler [2016] showed that averaging local estimators at the end will have bad dependence on either condition number or dimension of the problem. Yang [2013], Jaggi et al. [2014] and Smith et al. [2016] studied distributed optimization using stochastic (dual) coordinate descent, these approaches try to find a good balance between computation and communication, however, their communication complexity depends badly on the condition number. As a result, they are not better than first-order approaches, such as (proximal) accelerated gradient descent [Nesterov, 1983], in terms of communication. Shamir et al. [2014] and Zhang and Xiao [2015] proposed truly communication-efficient distributed optimization algorithms. They leveraged the local second-order information and, as a result, obtained milder dependence on the condition number compared to the first-order approaches [Boyd et al., 2011, Shamir and Srebro, 2014, Ma et al., 2015]. Lower bounds were studied in Zhang et al. [2013c], Braverman et al. [2016], and Arjevani and Shamir [2015]. However, it is not clear how to extend these existing approaches to problems with non-smooth objectives, including the ℓ_1 regularized problems.

Most of the above mentioned work is focused on estimators that are (asymptotically) linear. Averaging at the end reduces the variance of the these linear estimators, resulting in an estimator that matches the performance of a centralized procedure. Zhang et al. [2013d] studied averaging local estimators obtained by the penalized kernel ridge regression, with the ℓ_2 penalty was chosen smaller than usual to avoid the large bias problem. The situation in a high-dimensional setting is not so straightforward, since the sparsity inducing penalty introduces the bias in a non-linear way. Zhao et al. [2014b] illustrated how averaging debiased composite quantile regression estimators can be used for efficient inference in a high-dimensional setting. Averaging debiased high-dimensional estimators was subsequently used in Lee et al. [2017b] for distributed estimation, multi-task learning [Wang et al., 2016b], and statistical inference [Battey et al., 2015].

Chapter Organization. We describe our method in Section 2.2, and present the main results in the context of sparse linear regression in Section 2.3, and provide a generalized theory in Section 2.4. We demonstrate the effectiveness of the proposal via experiments in Section 7.5, and Section 2.6 contains detail proof of the technical results.

Notation. We use $[n]$ to denote the set $\{1, \dots, n\}$. For a vector $a \in \mathbb{R}^n$, we let $\text{support}(a) = \{j : a_j \neq 0\}$ be the support set, $||a||_q$, $q \in [1, \infty)$, the ℓ_q-norm defined as $||a||_q = (\sum_{i \in [n]} |a_i|^q)^{1/q}$, and $||\mathbf{a}||_\infty = \max_{i \in [n]} |a_i|$. For a matrix $A \in \mathbb{R}^{n_1 \times n_2}$, we use the following element-wise ℓ_∞ matrix norms $||A||_\infty = \max_{i \in [n_1], j \in [n_2]} |a_{ij}|$. Denote $\mathbf{I}_n$ as $n \times n$ identity matrix. For two sequences of numbers $\{a_n\}_{n=1}^\infty$ and $\{b_n\}_{n=1}^\infty$, we use $a_n = \mathcal{O}(b_n)$ to denote that $a_n \leq C b_n$ for some finite positive constant C, and for all n large enough. If $a_n = \mathcal{O}(b_n)$ and $b_n = \mathcal{O}(a_n)$, we use the notation $a_n \asymp b_n$. We also use $a_n \lesssim b_n$ for $a_n = \mathcal{O}(b_n)$ and $a_n \gtrsim b_n$ for $b_n = \mathcal{O}(a_n)$.

2.2 Methodology

In this section, we detail our procedure for estimating $\mathbf{w}^*$ in a distributed setting. Algorithm 1 provides an outline of the steps executed by the master and worker nodes. Let

$$\mathcal{L}_j(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_{ji}, \langle \mathbf{x}_{ji}, \mathbf{w} \rangle), \quad j \in [m],$$

be the *empirical* loss at each machine. Our method starts by solving a local ℓ_1 regularized M-estimation program. At iteration $t = 0$, the master (first) machine obtains $\widehat{\mathbf{w}}_0$ as a minimizer of the following program

$$\min \mathcal{L}_1(\mathbf{w}) + \lambda_0 ||\mathbf{w}||_1. \tag{2.3}$$

Algorithm 1 Efficient Distributed Sparse Learning (EDSL).

```
Input: Data $\{\mathbf{x}_{ji}, y_{ji}\}_{j\in[m],i\in[n]}$, loss function $\ell(\cdot,\cdot)$.
Initialization: The master obtains $\widehat{\mathbf{w}}_0$ by minimizing (6.2), and broadcast $\widehat{\mathbf{w}}_0$ to every worker.
for $t = 0, 1, \ldots$ do
    Workers:
    for $j = 2, 3, \ldots, m$ do
        if Receive $\widehat{\mathbf{w}}_t$ from the master then
            Calculate gradient $\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t)$ and send it to the master.
        end
    end
    Master:
    if Receive $\{\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t)\}_{j=2}^m$ from all workers then
        Obtain $\widehat{\mathbf{w}}_{t+1}$ by solving the shifted $\ell_1$ regularized problem in (2.4).
        Broadcast $\widehat{\mathbf{w}}_{t+1}$ to every worker.
    end
end
```

The vector $\widehat{\mathbf{w}}_0$ is broadcasted to all other machines, which use it to compute a gradient of the local loss at $\widehat{\mathbf{w}}_0$. In particular, each worker computes $\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_0)$ and communicates it back to the master. This constitutes one round of communication. At the iteration $t+1$, the master solves the shifted ℓ_1 regularized problem

$$\widehat{\mathbf{w}}_{t+1} = \arg\min_{\mathbf{w}} \mathcal{L}_1(\mathbf{w}) + \left\langle \frac{1}{m}\sum_{j=1}^m \nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t) - \nabla\mathcal{L}_1(\widehat{\mathbf{w}}_t), \mathbf{w} \right\rangle + \lambda_{t+1}||\mathbf{w}||_1. \tag{2.4}$$

A minimizer $\widehat{\mathbf{w}}_{t+1}$ is communicated to other machines, which use it to compute the local gradient $\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_{t+1})$ as before.

Formulation (2.4) is inspired by the proposal in Shamir et al. [2014], where the authors studied distributed optimization for smooth and strongly convex empirical objectives. Compared to Shamir et al. [2014], we do not use any averaging scheme, which would require additional rounds of communication and, moreover, we add an ℓ_1 regularization term to ensure consistent estimation in high-dimensions. Different from the distributed first-order optimization approaches, the refined objective (2.4) leverages both global first-order information and local higher-order information. To see this, suppose we set $\lambda_{t+1} = 0$ and that $\mathcal{L}_j(\mathbf{w})$ is a quadratic objective with invertible Hessian. Then we have the following closed form solution for (2.4),

$$\widehat{\mathbf{w}}_{t+1} = \widehat{\mathbf{w}}_t - \left(\nabla^2 \mathcal{L}_1(\widehat{\mathbf{w}}_t)\right)^{-1} \left(m^{-1} \sum_{j \in [m]} \nabla \mathcal{L}_j(\widehat{\mathbf{w}}_t) \right),$$

which is exactly a sub-sampled Newton updating rule. Unfortunately for high-dimensional problems, the Hessian is no longer invertible, and a ℓ_1 regularization is added to make the solution well behaved. The regularization parameter λ_t will be chosen in a way, so that it decreases with the iteration number t. As a result we will be able to show that the final estimator performs as well at the centralized solution. We discuss in details how to choose λ_t in the following section.

2.3 Main theoretical result

We illustrate our main theoretical results in the context of sparse linear regression model

$$y_{ji} = \langle \mathbf{x}_{ji}, \mathbf{w}^* \rangle + \epsilon_{ji}, \qquad i \in [n], j \in [m], \tag{2.5}$$

where $\mathbf{x}_{ji}$ is a subgaussian p-dimensional vector of input variables and ϵ_{ji} is i.i.d. mean zero subgaussian noise. The loss function considered is the usual the squared loss $\ell(y, \widehat{y}) = \frac{1}{2}(y - \widehat{y})^2$. With this notation, the centralized approach leads to the lasso estimator [Tibshirani, 1996]

$$\widehat{\mathbf{w}}_{\text{centralize}} = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{j=1}^{m} \mathcal{L}_j(\mathbf{w}) + \lambda ||\mathbf{w}||_1,$$

where the loss at worker j is

$$\mathcal{L}_j(\mathbf{w}) = \frac{1}{2n} \sum_{i \in [n]} (y_{ji} - \langle \mathbf{w}, \mathbf{x}_{ji} \rangle)^2.$$

Before stating the main result, we provide the definition of the subgaussian norm [Vershynin, 2010].

Definition 1 (Subgaussian norm). *The subgaussian norm $||X||_{\psi_2}$ of a subgaussian p-dimensional random vector X, is defined as*

$$||X||_{\psi_2} = \sup_{x \in \mathbb{S}^{p-1}} \sup_{q > 1} q^{-1/2} (\mathbb{E}|\langle X, x \rangle|^q)^{1/q},$$

where $\mathbb{S}^{p-1}$ is the p-dimensional unit sphere.

We also need an assumption on the restricted strong convexity constant [Negahban et al., 2012].

Assumption 2. *We assume that there exists a $\kappa > 0$, such that for any $\Delta \in \mathcal{C}(S, 3)$,*

$$\frac{1}{2n} ||\mathbf{X}_1 \Delta||_2^2 \geq \kappa ||\Delta||_2^2,$$

where

$$\mathcal{C}(S, 3) = \{\Delta \in \mathbb{R}^p \mid ||\Delta_{S^c}||_1 \leq 3||\Delta_S||_1\}$$

is a restricted cone in $\mathbb{R}^p$, and

$$\mathbf{X}_1 = [\mathbf{x}_{11}^T; \mathbf{x}_{12}^T; \ldots; \mathbf{x}_{1n}^T] \in \mathbb{R}^{n \times p}$$

is the data matrix on the master machine.

When $\mathbf{x}_{ji}$ are randomly drawn from a subgaussian distribution, Assumption (2) is satisfied with high probability as long as $n \gtrsim s \log p$ [Rudelson and Zhou, 2013].

We are now ready to state the estimation error bound for $\widehat{\mathbf{w}}_{t+1}$ obtained using Algorithm 1.

Theorem 3. *Assume that data are generated from a sparse linear regression model in* (2.5) *with* $||\mathbf{x}_{ji}||_{\psi_2} \leq \sigma_X$ *and* $||\epsilon_{ji}||_{\psi_2} \leq \sigma$. *Let*

$$\lambda_{t+1} = \frac{2}{mn}\Bigg|\Bigg| \sum_{j\in[m]} \sum_{i\in[n]} \mathbf{x}_{ji}\epsilon_{ji} \Bigg|\Bigg|_\infty + 2L\left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^2\right) \cdot ||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1 \cdot \sqrt{\frac{\log(2p/\delta)}{n}} \tag{2.6}$$

Then for $t \geq 0$ *we have, with probability at least* $1 - 2\delta$*,*

$$||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 \leq \frac{1 - a_n^{t+1}}{1 - a_n} \frac{48 s\sigma\sigma_X}{\kappa} \sqrt{\frac{\log(p/\delta)}{mn}} + a_n^{t+1} \frac{s\sigma\sigma_X}{\kappa} \sqrt{\frac{\log(np/\delta)}{n}}, \tag{2.7}$$

$$||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2 \leq \frac{1 - a_n^{t+1}}{1 - a_n} \frac{12\sqrt{s}\sigma\sigma_X}{\kappa} \sqrt{\frac{\log(p/\delta)}{mn}} + a_n^{t} b_n \frac{s\sigma\sigma_X}{\kappa} \sqrt{\frac{\log(np/\delta)}{n}}, \tag{2.8}$$

where

$$a_n = \frac{96 s\sigma\sigma_X}{\kappa} \sqrt{\frac{\log(2p/\delta)}{n}}$$

and

$$b_n = \frac{24\sqrt{s}\sigma\sigma_X}{\kappa} \sqrt{\frac{\log(np/\delta)}{n}}.$$

We can simplify the bound obtained in Theorem 3 by looking at the scaling with respect to n, m, s, and p, by treating κ, σ and σ_X as constants. Suppose $n \gtrsim s^2 \log p$ and set

$$\lambda_t \asymp \sqrt{\frac{\log p}{mn}} + \sqrt{\frac{\log p}{n}} \left(s\sqrt{\frac{\log p}{n}}\right)^t.$$

The following error bounds hold for Algorithm 1:

$$||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1 \lesssim_P s\sqrt{\frac{\log p}{mn}} + \left(s\sqrt{\frac{\log p}{n}}\right)^{t+1},$$
$$||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_2 \lesssim_P \sqrt{\frac{s\log p}{mn}} + \left(\sqrt{\frac{s\log p}{n}}\right)\left(s\sqrt{\frac{\log p}{n}}\right)^{t}.$$

We can compare the above bounds to the performance of the local and centralized lasso [Wainwright, 2009, Meinshausen and Yu, 2009, Bickel et al., 2009]. For $\widehat{\mathbf{w}}_{\text{local}}$, we have

$$||\widehat{\mathbf{w}}_{\text{local}} - \mathbf{w}^*||_1 \lesssim_P s\sqrt{\frac{\log p}{n}}$$

and

$$||\widehat{\mathbf{w}}_{\text{local}} - \mathbf{w}^*||_2 \lesssim_P \sqrt{\frac{s\log p}{n}}.$$

For $\widehat{\mathbf{w}}_{\text{centralize}}$, we have

$$||\widehat{\mathbf{w}}_{\text{centralize}} - \mathbf{w}^*||_1 \lesssim_P s\sqrt{\frac{\log p}{mn}}$$

and

$$||\widehat{\mathbf{w}}_{\text{centralize}} - \mathbf{w}^*||_2 \lesssim_P \sqrt{\frac{s\log p}{mn}}.$$

We see that after one round of communication, we have

$$||\widehat{\mathbf{w}}_1 - \mathbf{w}^*||_1 \lesssim_P s\sqrt{\frac{\log p}{mn}} + \frac{s^2\log p}{n}$$

and

$$||\widehat{\mathbf{w}}_1 - \mathbf{w}^*||_2 \lesssim_P \sqrt{\frac{s\log p}{mn}} + \frac{s^{3/2}\log p}{n}.$$

These bounds match the results in Lee et al. [2017b] without expensive debiasing step. Furthermore, when $m \lesssim \frac{n}{s^2\log p}$, they match the performance of the centralized lasso. Finally, as long as $t \gtrsim \log m$ and $n \gtrsim s^2 \log p$, it is easy to check that $\left(s\sqrt{\frac{\log p}{n}}\right)^{t+1} \lesssim s\sqrt{\frac{\log p}{mn}}$. Therefore,

$$||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 \lesssim_P s\sqrt{\frac{\log p}{mn}}$$

and

$$||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2 \lesssim_P \sqrt{\frac{s \log p}{mn}},$$

which matches the centralized lasso performance without additional error terms. That is, as long as $n \gtrsim s^2 \log p$, the rounds of communication to matches centralized procedure only increase logarithmically with the number of machines and independent of other parameters. Differently, for distributed learning methods studied in the literature for minimizing smooth objectives, the rounds of communication to match centralized procedure increase polynomially with m (see table 1 in Zhang and Xiao [2015]). This is because here we exploit the underlying restricted strong convexity from empirical loss functions, while prior work on distributed minimization of smooth objectives Shamir et al. [2014], Zhang and Xiao [2015] only consider strong convexity explicitly from regularization.

2.4 Generalized theory and proof sketch

In order to establish Theorem 3, we prove an error bound on $\widehat{\mathbf{w}} - \mathbf{w}^*$ for a general loss $\ell(\cdot, \cdot)$ and $\widehat{\mathbf{w}}$ obtained using Algorithm 1. To simplify the presentation, we assume that the domain $\mathcal{X}$ is bounded and that the loss function $\ell(\cdot, \cdot)$ is smooth.

Assumption 4. *The loss $\ell(\cdot, \cdot)$ is L-smooth with respect to the second argument:*

$$\ell'(a, b) - \ell'(a, c) \leq L|b - c|, \qquad \forall a, b, c \in \mathbb{R}$$

Furthermore, $|\ell'''(a, b)| \leq M$ for all $a, b \in \mathbb{R}$.

Commonly used loss functions in statistical learning, including the squared loss for regression and logistic loss for classification, satisfy this assumption [Zhang et al., 2013g].

Next, we state the restricted strong convexity condition for a general loss function [Negahban et al., 2012].

Assumption 5. *There exists* $\kappa > 0$ *such that for any* $\Delta \in \mathcal{C}(S, 3)$

$$\mathcal{L}_1(\mathbf{w}^* + \Delta) - \mathcal{L}_1(\mathbf{w}^*) - \langle \nabla \mathcal{L}_1(\mathbf{w}^*), \Delta \rangle \geq \kappa ||\Delta||_2^2,$$

with $\mathcal{C}(S, 3) = \{\Delta \in \mathbb{R}^p | ||\Delta_{S^c}||_1 \leq 3||\Delta_S||_1\}$.

The restricted strong convexity holds with high probability for a wide range of models and designs and it is commonly assumed for showing consistent estimation in high-dimensions [see, for example, Van De Geer et al., 2009, Negahban et al., 2012, Raskutti et al., 2010, Rudelson and Zhou, 2013, for details].

Our main theoretical result establishes a recursive estimation error bound, which relates the estimation error $||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||$ to that of the previous iteration $||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1$.

Theorem 6. *Suppose Assumption 4 and 5 holds. Let*

$$\begin{aligned} \lambda_{t+1} =& 2\left|\left|\frac{1}{m}\sum_{j\in[m]} \nabla \mathcal{L}_j(\mathbf{w}^*)\right|\right|_\infty + 2L\left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^2\right)||\mathbf{w}^* - \widehat{\mathbf{w}}_t||_1 \sqrt{\frac{\log(2p/\delta)}{n}} \\ &+ 2M\left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^3\right)\left(||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1^2\right). \end{aligned} \tag{2.9}$$

Then with probability at least $1 - \delta$*, we have*

$$\begin{aligned} ||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 \leq& \frac{48s}{\kappa}\left|\left|\frac{1}{m}\sum_{j\in[m]} \nabla \mathcal{L}_j(\mathbf{w}^*)\right|\right|_\infty + \frac{48sL}{\kappa}\left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^2\right)||\mathbf{w}^* - \widehat{\mathbf{w}}_t||_1 \sqrt{\frac{\log(2p/\delta)}{n}} \\ &+ \frac{48sM}{\kappa}\left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^3\right)\left(||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1^2\right), \end{aligned}$$

and

$$||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2 \leq \frac{12\sqrt{s}}{\kappa} \left|\left| \frac{1}{m} \sum_{j \in [m]} \nabla \mathcal{L}_j(\mathbf{w}^*) \right|\right|_\infty + \frac{12\sqrt{s}L}{\kappa} \left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^2 \right) ||\mathbf{w}^* - \widehat{\mathbf{w}}_t||_1 \sqrt{\frac{\log(2p/\delta)}{n}}$$
$$+ \frac{4\sqrt{s}M}{\kappa} \left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^3 \right) \left(||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1^2 \right).$$

Theorem 6 upper bounds the estimation error $||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1$ as a function of $||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1$. Applying Theorem 6 iteratively, we immediately obtain the following estimation error bound which depends on the quality of local ℓ_1 regularized estimation $||\widehat{\mathbf{w}}_0 - \mathbf{w}^*||_1$.

Corollary 7. *Suppose the conditions of Theorem 6 are satisfied. Furthermore, suppose that for all t, we have*

$$M \left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty \right) ||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1 \leq L \sqrt{\frac{\log(2p/\delta)}{n}}. \tag{2.10}$$

Then with probability at least $1 - \delta$, we have

$$||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 \leq a_n^{t+1} ||\widehat{\mathbf{w}}_0 - \mathbf{w}^*||_1 + (1 - a_n)^{-1}(1 - a_n^{t+1}) \cdot \frac{48s}{\kappa} \cdot \left|\left| \frac{1}{m} \sum_{j \in [m]} \nabla \mathcal{L}_j(\mathbf{w}^*) \right|\right|_\infty$$

and

$$||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2 \leq a_n^t b_n \cdot ||\widehat{\mathbf{w}}_0 - \mathbf{w}^*||_1 + (1 - a_n)^{-1}(1 - a_n^{t+1}) \cdot \frac{12\sqrt{s}}{\kappa} \cdot \left|\left| \frac{1}{m} \sum_{j \in [m]} \nabla \mathcal{L}_j(\mathbf{w}^*) \right|\right|_\infty,$$

where

$$a_n = \frac{96sL}{\kappa} \left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^2 \right) \sqrt{\frac{\log(2p/\delta)}{n}}$$

and

$$b_n = \frac{24\sqrt{s}L}{\kappa} \left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^2 \right) \sqrt{\frac{\log(2p/\delta)}{n}}.$$

For the quadratic loss we have that $M = 0$ and the condition in (2.10) holds. For other types of losses, condition in (2.10) will be true for t large enough when $m \gtrsim s^2$, leading to local exponential rate of convergence until reaching statistical optimal region.

2.4.1 Proof sketch of Theorem 6

We first analyze how the estimation error bound decreases after one round of communication. In particular, we bound $||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||$ with $||\widehat{\mathbf{w}}_t - \mathbf{w}^*||$. Define

$$\widetilde{\mathcal{L}}_1(\mathbf{w}, \widehat{\mathbf{w}}_t) = \mathcal{L}_1(\mathbf{w}) + \left\langle \frac{1}{m} \sum_{j\in[m]} \nabla \mathcal{L}_j(\widehat{\mathbf{w}}_t) - \nabla \mathcal{L}_1(\widehat{\mathbf{w}}_t), \mathbf{w} \right\rangle. \tag{2.11}$$

Then

$$\nabla \widetilde{\mathcal{L}}_1(\mathbf{w}, \widehat{\mathbf{w}}_t) = \nabla \mathcal{L}_1(\mathbf{w}) + \frac{1}{m} \sum_{j\in[m]} \nabla \mathcal{L}_j(\widehat{\mathbf{w}}_t) - \nabla \mathcal{L}_1(\widehat{\mathbf{w}}_t).$$

The following lemma bounds the ℓ_∞ norm of $\nabla \widetilde{\mathcal{L}}_1(\mathbf{w}, \widehat{\mathbf{w}}_t)$.

Lemma 8. *With probability at least $1 - \delta$, we have*

$$\begin{aligned} \left|\left|\nabla \widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t)\right|\right|_\infty \leq & \left|\left|\frac{1}{m} \sum_{j\in[m]} \nabla \mathcal{L}_j(\mathbf{w}^*)\right|\right|_\infty + 2L \left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^2\right) ||\mathbf{w}^* - \widehat{\mathbf{w}}_t||_1 \sqrt{\frac{\log(2p/\delta)}{n}} \\ & + M \left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^3\right) \left(||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1^2\right). \end{aligned}$$

The lemma bounds the magnitude of the gradient of the loss at optimum point $\mathbf{w}^*$. This will be used to guide our choice of the ℓ_1 regularization parameter λ_{t+1} in (2.4). The following lemma shows that as long as λ_{t+1} is large enough, it is guaranteed that $\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*$ is in a restricted cone.

Lemma 9. *Suppose*

$$\lambda_{t+1}/2 \geq \left|\left|\nabla \widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t)\right|\right|_\infty.$$

Then with probability at least $1-\delta$, *we have* $\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^* \in \mathcal{C}(S, 3)$.

Based on the conic condition and restricted strong convexity condition, we can obtain the recursive error bound stated in Theorem 6 following the proof strategy as in Negahban et al. [2012].

2.4.2 Application example: sparse logistic regression

For logistic model, performing maximum likelihood estimation (MLE) on the following logistic model:

$$\mathbb{P}(y_{ji} = \pm 1 | \mathbf{x}_{ji}) = \frac{\exp(y_{ji}\langle \mathbf{x}_{ji}, \mathbf{w}^*\rangle)}{\exp(y_{ji}\langle \mathbf{x}_{ji}, \mathbf{w}^*\rangle) + 1}, \tag{2.12}$$

which leads to the logistic loss function $\ell(y_{ji}, \langle \mathbf{w}, \mathbf{x}_{ji}\rangle) = \log(1 + \exp(-y_{ji}\langle \mathbf{w}, \mathbf{x}_{ji}\rangle))$. For high-dimensional problems, when we add a ℓ_1 regularization, we obtain the ℓ_1 regularized logistic regression model [Zhu and Hastie, 2004, Wu et al., 2009]:

$$\widehat{\mathbf{w}}_{\text{centralize}} = \arg\min_{\mathbf{w}} \frac{1}{mn} \sum_{j\in[m]} \sum_{i\in[n]} \log(1 + \exp(-y_{ji}\langle \mathbf{w}, \mathbf{x}_{ji}\rangle)) + \lambda ||\mathbf{w}||_1.$$

The logistic loss is $\frac{1}{4}$-smooth, and we also know $M = \frac{1}{4}$ because of self-concordance [Zhang and Xiao, 2015]. Let $\mathcal{L}_j(\mathbf{w}) = \frac{1}{n}\sum_{i\in[n]} \log(1 + \exp(-\mathbf{y}_{ji}\langle \mathbf{w}, \mathbf{x}_{ji}\rangle))$, [Negahban et al., 2012] showed that if $\mathbf{x}_{ji}$ are drawn from mean zero distribution with sub-Gaussian tails, then $\mathcal{L}_1(\mathbf{w})$ satisfies the restricted strong condition (5). Moreover, we have the following control on the quantity $\left|\left|\frac{1}{m}\sum_{j\in[m]} \nabla \mathcal{L}_j(\mathbf{w}^*)\right|\right|_\infty$.

Lemma 10. *Then we have the following upper bound holds in probability at least* $1-\delta$*:*

$$\left\|\frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\mathbf{w}^*)\right\|_\infty \lesssim ||\mathbf{x}_{ji}||_\infty\sqrt{\frac{2\log(p/\delta)}{mn}}.$$

The following ℓ_1 error bound states the estimation error for logistic regression with ℓ_1 regularization, which was established, for example, in [Van de Geer et al., 2008, Negahban et al., 2012].

Lemma 11. *Under the model* (2.12)*, when* $n \geq (64/\kappa)s\log p$*, we have the following estimation error bound for* $\widehat{\mathbf{w}}_0$ *holds with probability at least* $1-\delta$*:*

$$||\widehat{\mathbf{w}}_0-\mathbf{w}^*||_1 \lesssim \frac{s\sigma_X}{\kappa}\sqrt{\frac{2\log(np/\delta)}{n}}.$$

With above analysis for sparse logistic regression model with random design, we are ready to present the results for the estimation error bound which established local exponential convergence.

Corollary 12. *Under sparse logistic regression model with random design, and set* λ_{t+1} *as* (2.9)*. If the following condition holds for some* $T \geq 0$*:*

$$||\widehat{\mathbf{w}}_T-\mathbf{w}^*||_1 \leq 4\sqrt{\frac{\log(2p/\delta)}{n}}. \tag{2.13}$$

Then with probability at least $1-2\delta$*, we have the following estimation error bound for all* $t \geq T$*:*

$$||\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*||_1 \leq \frac{1-a_n^{t-T+1}}{1-a_n}\frac{96s\sigma\sigma_X}{\kappa}\sqrt{\frac{\log(p/\delta)}{mn}}+4a_n^{t-T+1}\sqrt{\frac{\log(2p/\delta)}{n}}, \tag{2.14}$$

$$||\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*||_2 \leq \frac{1-a_n^{t-T+1}}{1-a_n}\frac{4\sqrt{s}\sigma\sigma_X}{\kappa}\sqrt{\frac{\log(p/\delta)}{mn}}+4a_n^{t-T}b_n\sqrt{\frac{\log(2p/\delta)}{n}}, \tag{2.15}$$

where

$$a_n = \frac{24 s \sigma \sigma_X}{\kappa} \sqrt{\frac{\log(2p/\delta)}{n}} \quad \textit{and} \quad b_n = \frac{\sqrt{s} \sigma \sigma_X}{\kappa} \sqrt{\frac{\log(np/\delta)}{n}}.$$

2.4.3 Application example: high-dimensional generalized linear models

The results are readily extendable to other high-dimensional generalized linear models [McCullagh and Nelder, 1989, Van de Geer et al., 2008], where the response variable $y_{ji} \in \mathcal{Y}$ is drawn from the distribution

$$\mathbb{P}(y_{ji}|\mathbf{x}_{ji}) \propto \exp\left(\frac{y_{ji}\langle \mathbf{x}_{ji}, \mathbf{w}^* \rangle - \Phi(\langle \mathbf{x}_{ji}, \mathbf{w}^* \rangle)}{A(\sigma)} \right),$$

where $\Phi(\cdot)$ is a link function and $A(\sigma)$ is a scale parameter. Under the random subgaussian design, as long as the loss function has Lipschitz gradient, then the algorithm and corresponding estimation error bound and be applied.

2.4.4 Application example: high-dimensional graphical models

The results can also be used for the distributed unsupervised learning setting where the task is to learn a sparse graphical structure that represents the conditional independence between variables. Widely studied graphical models are Gaussian graphical models [Meinshausen and Bühlmann, 2006, Yuan and Lin, 2007] for continuous data and Ising graphical models [Ravikumar et al., 2010] for binary observations. As shown in [Meinshausen and Bühlmann, 2006, Ravikumar et al., 2010], these model selection problems can be reduced to solving parallel ℓ_1 regularized linear regression and logistic regression problems, respectively. Thus the approach presented in this chapter can be readily applicable for these tasks.

2.5 Experiments

In this section we present empirical comparisons between various approaches on both simulated and real world datasets. We run the algorithms for both distributed regression and classification problems, and compare with the following algorithms: i) Local; ii) Centralize; iii) Distributed proximal gradient descent (Prox GD); iv) Avg-Debias [Lee et al., 2017b] with hard thresholding, and v) the proposed EDSL approach.

2.5.1 Simulations

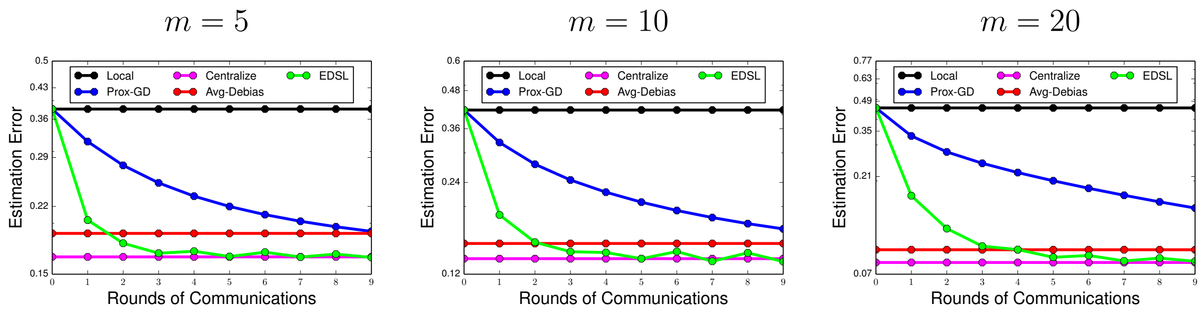
We first examine the algorithms on simulated data. We generate $\{\mathbf{x}_{ji}\}_{j\in[m],i\in[n]}$ from a multivariate normal distribution with mean zero and covariance matrix $\mathbf{\Sigma}$. The covariance $\mathbf{\Sigma}$ controls the condition number of the problem and we will varying it to see how the performance changes. We set $\Sigma_{ij} = 0.5^{|i-j|}$ for the well-conditioned setting and $\Sigma_{ij} = 0.5^{|i-j|/5}$ for the ill-conditioned setting. The response variable $\{y_{ji}\}_{j\in[m],i\in[n]}$ are drawn from (2.5) and (2.12) for regression and classification problems, respectively. For regression, the noise ϵ_{ji} is sampled from a standard normal distribution. The true model $\mathbf{w}^*$ is set to be s-sparse, where the first s-entries are sampled i.i.d. from a uniform distribution in $[0, 1]$, and the other entries are set to zero.

We run experiments with various (n, p, m, s) settings[3]. The estimation error $||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_2$ is shown versus rounds of communications for for Prox GD and the proposed EDSL algorithm. We also plot the estimation error of Local, Avg-Debias, and Centralize as horizontal lines, since the communication cost is fixed for for these algorithms[4][5]. Figure 2.1 and 2.2 summarize the results, averaged across 10 independent trials. We have the following observations:

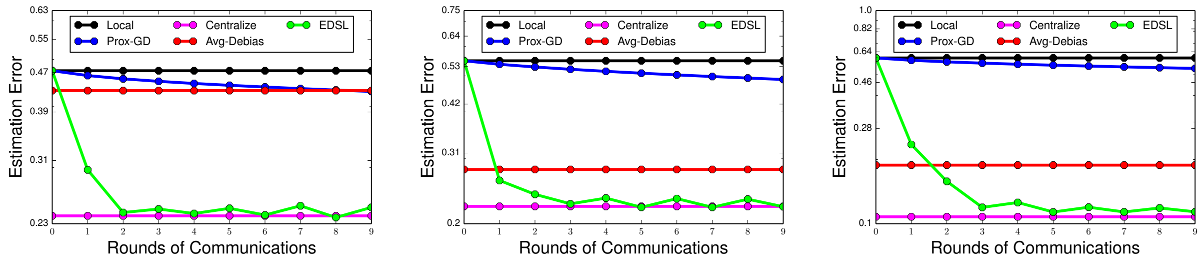
3. n: sample size per machine, p: problem dimension, m: number of machines, s: true support size.

4. these algorithms have zero, one-shot and full communications, respectively.

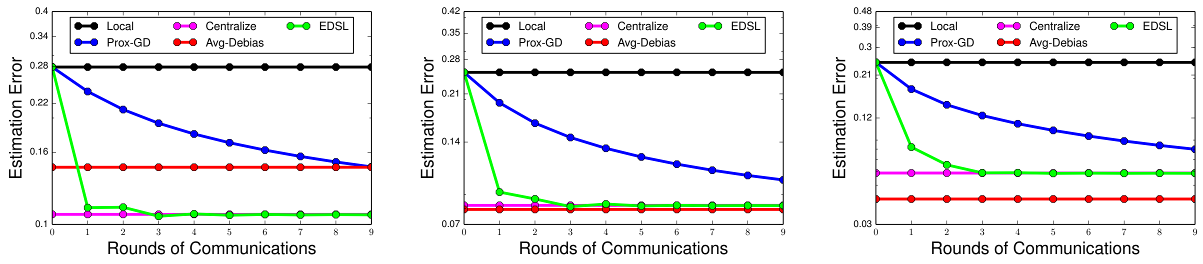
5. To give some senses about computational cost, for a problem with $n = 200, p = 1000$, at each round EDSL takes about 0.048s, while Avg-Debias takes about 40.334s.



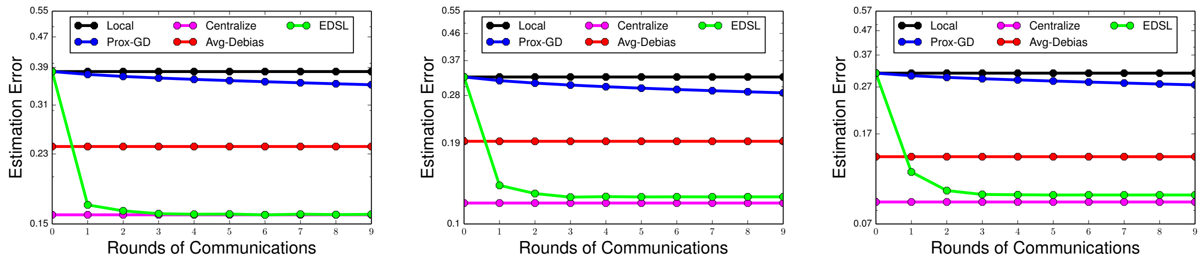
$$n = 200, p = 1000, s = 10, \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \mathbf{\Sigma}_{ij} = 0.5^{|i-j|}.$$



$$n = 200, p = 1000, s = 10, \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \mathbf{\Sigma}_{ij} = 0.5^{|i-j|/5}.$$



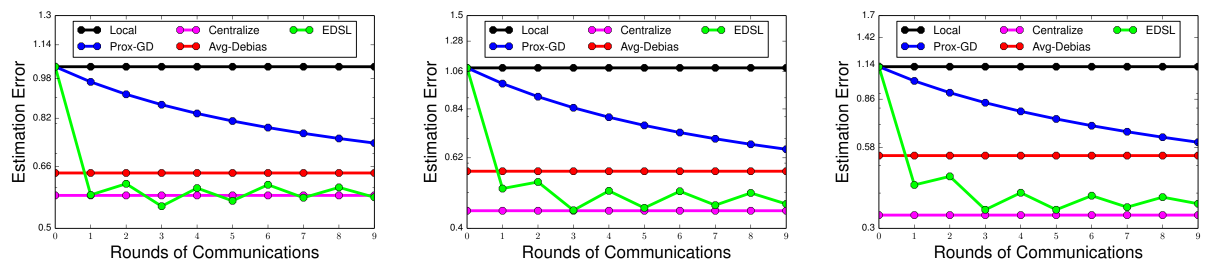
$$n = 500, p = 3000, s = 10, \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \mathbf{\Sigma}_{ij} = 0.5^{|i-j|}.$$



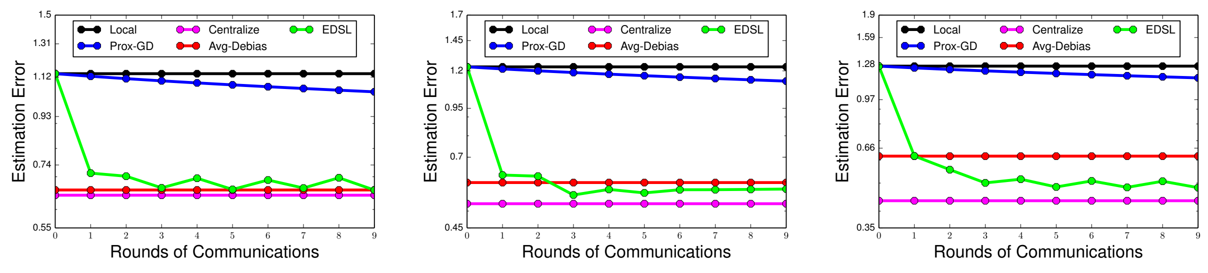
$$n = 500, p = 3000, s = 10, \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \mathbf{\Sigma}_{ij} = 0.5^{|i-j|/5}.$$

Figure 2.1: Comparison of various algorithms for distributed sparse regression, 1st and 3rd row: well-conditioned cases, 2nd and 4th row: ill-conditioned cases.

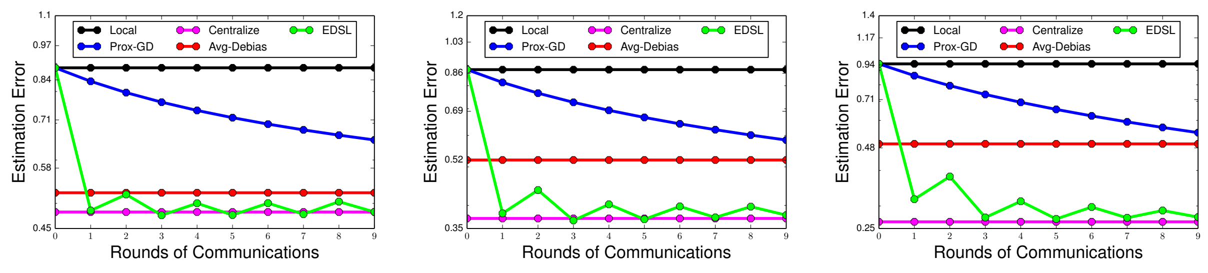
$m = 5$ $\qquad$ $m = 10$ $\qquad$ $m = 20$



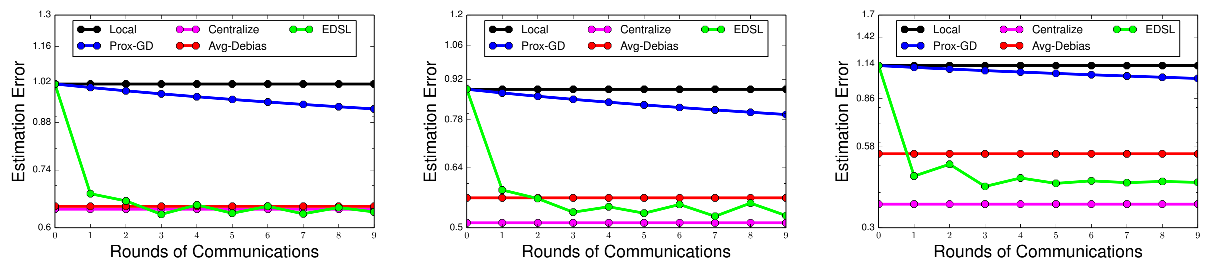
$n = 500, p = 1000, s = 10, \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \mathbf{\Sigma}_{ij} = 0.5^{|i-j|}$.



$n = 500, p = 1000, s = 10, \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \mathbf{\Sigma}_{ij} = 0.5^{|i-j|/5}$.



$n = 1000, p = 3000, s = 10, \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \mathbf{\Sigma}_{ij} = 0.5^{|i-j|}$.



$n = 1000, p = 3000, s = 10, \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \mathbf{\Sigma}_{ij} = 0.5^{|i-j|/5}$.

Figure 2.2: Comparison of various algorithms for distributed sparse classification (logistic regression), 1st and 3rd row: well-conditioned cases, 2nd and 4th row: ill-conditioned cases.

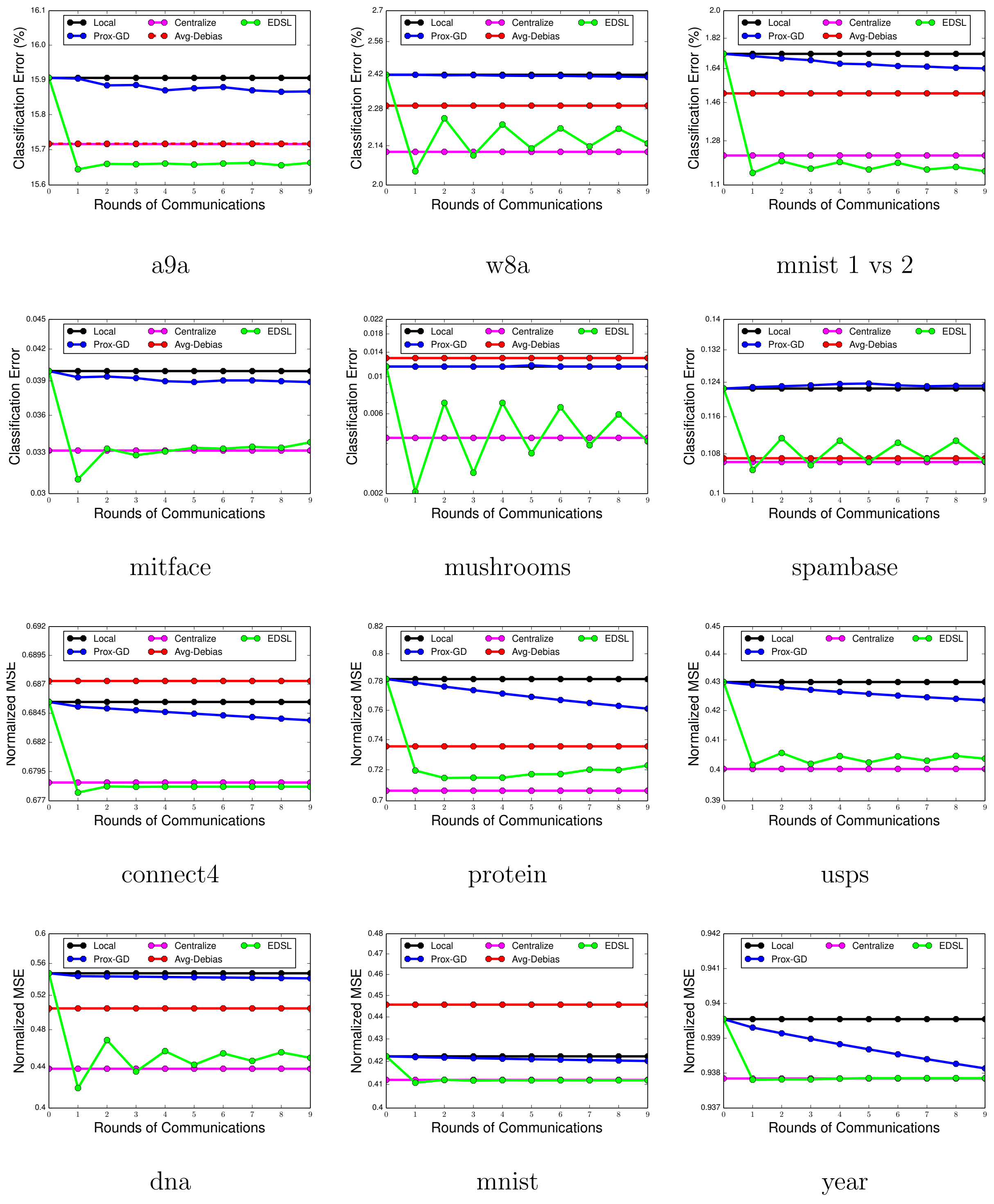


Figure 2.3: Comparison of various approaches for distributed sparse regression and classification on real world datasets. (Avg-Debias is omitted when it is significantly worse than others.)

- The Avg-Debias approach obtained much better estimation error compared to Local after one round of communication and sometimes performed quite close to Centralize. However, in most cases, there is still a gap compared with Centralize, especially when the problem is not well-conditioned or m is large.

- ProxGD converges very slow when the condition number becomes bad ($\Sigma_{ij} = 0.5^{|i-j|/5}$ case).

- As theory suggests, EDSL obtained a solution that is competitive with Avg-Debias after one round of communication. The estimation error decreases to match performance of Centralize within few rounds of communications; typically less than 5, even though the theory suggests EDSL will match the performance of centralize within $\mathcal{O}(\log m)$ rounds of communication.

Above experiments illustrate our theoretical results in finite samples. As suggested by theory, when sample size per machine n is relatively small, one round of communication is not sufficient to make Avg-Debias matches the performance of centralized procedure. However, EDSL could match the performance of Avg-Debias with one round of communication and further improve the estimation quality by exponentially reducing the gap between centralized procedure with Avg-Debias, until matching the centralized performance. Thus, the proposed EDSL improves the Avg-Debias approach both computationally and statistically.

2.5.2 Real-world data evaluation

In this section, we compare the distributed sparse learning algorithms on several real world datasets. The datasets are publicly available from the LIBSVM website[6] and UCI Machine Learning Repository[7]. The statistics of these datasets are summarized in Table 5.2, where

6. https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

7. http://archive.ics.uci.edu/ml/

Table 2.2: List of real-world datasets used in the experiments.

Name	#Instances	#Features	Task
a9a	48,842	123	Classification
connect-4	67,557	127	Regression
dna	2,000	181	Regression
mitface	6,977	362	Classification
mnist 1 vs 2	14,867	785	Classification
mnist	60,000	785	Regression
mushrooms	8,124	113	Classification
protein	17,766	358	Regression
spambase	4,601	57	Classification
usps	7,291	257	Regression
w8a	64,700	301	Classification
year	51,630	91	Regression

some of the multi-class classification datasets are adopted under the regression setting with squared losses. For all data sets, we use 60% of data for training, 20% as held-out validation set for tuning the parameters, and the remaining 20% for testing. We randomly partition data 10 times and report the average performance on the test set. For regression tasks, the evaluation metric is the normalized Mean Squared Error (normalized MSE), while for classification tasks we report the miss-classification error. We randomly partition the data on $m = 10$ machines. The results are plotted in Figure 7.2 where for some datasets the performance of Avg-Debias is significantly worse than others (mostly because the debiasing step fails), thus we omit these plots. The plots are shown in Figure 7.2 We have the following observations:

- Since there is no well-specified model on these datasets, the curves behave quite differently on different data sets. However, a large gap between the local and centralized procedure is consistent as the later uses 10 times more data.

- Avg-Debias often fails on these real datasets and performs much worse than in simulations. The main reason might be that the assumptions, such as well-specified model or generalized coherence condition, fail, then Avg-Debias can totally fail and produce solution even much worse than the local.

- Prox GD approach still converges slowly in most of the cases.
- The proposed EDSL is quite robust on real world data sets, and can output a solution which is highly competitive with the centralized model within a few rounds of communications.
- There exits a slight "zig-zag" behavior for EDSL approach on some data sets. For example, on the mushrooms data set, the predictive performance of EDSL is not stable.

In sum, the experimental results on real world data sets verified that the proposed EDSL method is effective for distributed sparse learning problems.

2.6 Proofs of technical results

The section contains proofs of some theorems and lemmas stated in this chapter.

2.6.1 Proof of Lemma 8

Proof. Recall the definition of $\widetilde{\mathcal{L}}_1$ from (2.11). We have

$$\begin{aligned}\nabla\widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t) =& \nabla\mathcal{L}_1(\mathbf{w}^*) + \frac{1}{m}\sum_{j\in[m]} \nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t) - \nabla\mathcal{L}_1(\widehat{\mathbf{w}}_t) \\ =& \frac{1}{m}\sum_{j\in[m]} \nabla\mathcal{L}_j(\mathbf{w}^*) + \nabla\mathcal{L}_1(\mathbf{w}^*) - \nabla\mathcal{L}_1(\widehat{\mathbf{w}}_t) \\ & - \left(\frac{1}{m}\sum_{j\in[m]} \nabla\mathcal{L}_j(\mathbf{w}^*) - \frac{1}{m}\sum_{j\in[m]} \nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t)\right).\end{aligned}$$

Using the triangle inequality

$$\left\|\nabla\widetilde{\mathcal{L}}_1(\mathbf{w}^*,\widehat{\mathbf{w}}_t)\right\|_\infty \leq \left\|\frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\mathbf{w}^*)\right\|_\infty + \left\|\nabla\mathcal{L}_1(\mathbf{w}^*) - \nabla\mathcal{L}_1(\widehat{\mathbf{w}}_t) - \left(\frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\mathbf{w}^*) - \frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t)\right)\right\|_\infty.$$

We focus on bounding the second term in the right-hand-side inequality above. Let $\tau_{ji} = \ell'(y_{ji},\langle\mathbf{w}^*,\mathbf{x}_{ji}\rangle)$ and define $\mathbf{v}_{ji}(\widehat{\mathbf{w}}_t)\in\mathbb{R}^p$:

$$\begin{aligned}\mathbf{v}_{ji}(\widehat{\mathbf{w}}_t) &= \mathbf{x}_{ji}(\ell'(y_{ji},\langle\mathbf{w}^*,\mathbf{x}_{ji}\rangle) - \ell'(y_{ji},\langle\widehat{\mathbf{w}}_t,\mathbf{x}_{ji}\rangle))\\ &= \tau_{ji}\mathbf{x}_{ji}\mathbf{x}_{ji}^T(\widehat{\mathbf{w}}_t - \mathbf{w}^*) + \mathbf{x}_{ji}\frac{\ell'''(y_{ji},\mathbf{u}_{ji})}{2}(\langle\widehat{\mathbf{w}}_t - \mathbf{w}^*,\mathbf{x}_{ji}\rangle)^2\end{aligned}$$

where $\mathbf{u}_{ji}$ is a number between $\langle\widehat{\mathbf{w}}_t,\mathbf{x}_{ji}\rangle$ and $\langle\mathbf{w}^*,\mathbf{x}_{ji}\rangle$. With this notation

$$\begin{aligned}&\left\|\nabla\mathcal{L}_1(\mathbf{w}^*) - \nabla\mathcal{L}_1(\widehat{\mathbf{w}}_t) - \left(\frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\mathbf{w}^*) - \frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t)\right)\right\|_\infty\\ &\leq \left\|\frac{1}{n}\sum_{i\in[n]}\mathbf{v}_{1i}(\widehat{\mathbf{w}}_t) - \frac{1}{mn}\sum_j\sum_i\mathbf{v}_{ji}(\widehat{\mathbf{w}}_t)\right\|_\infty\\ &\leq \left\|\frac{1}{n}\sum_i\tau_{1i}\mathbf{x}_{1i}\mathbf{x}_{1i}^T(\widehat{\mathbf{w}}_t - \mathbf{w}^*) - \frac{1}{mn}\sum_j\sum_i\tau_{ji}\mathbf{x}_{ji}\mathbf{x}_{ji}^T(\widehat{\mathbf{w}}_t - \mathbf{w}^*)\right\|_\infty\\ &\quad + M\cdot\left(\max_{j,i}||\mathbf{x}_{ji}||_\infty^3\right)\cdot||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1^2\,.\end{aligned}$$

The first term above can be further upper bounded by

$$\begin{aligned}
&\left|\left|\frac{1}{n}\sum_{j}\tau_{1i}\mathbf{x}_{1i}\mathbf{x}_{1i}^T(\widehat{\mathbf{w}}_t-\mathbf{w}^*)-\frac{1}{mn}\sum_{j}\sum_{i}\tau_{ji}\mathbf{x}_{ji}\mathbf{x}_{ji}^T(\widehat{\mathbf{w}}_t-\mathbf{w}^*)\right|\right|_\infty \\
&\quad\le\left|\left|\frac{1}{n}\sum_{j}\tau_{1i}\mathbf{x}_{1i}\mathbf{x}_{1i}^T-\frac{1}{mn}\sum_{j}\sum_{i}\tau_{ji}\mathbf{x}_{ji}\mathbf{x}_{ji}^T\right|\right|_\infty\cdot||\widehat{\mathbf{w}}_t-\mathbf{w}^*||_1\ . \\
&\quad\le\left(\left|\left|\frac{1}{n}\sum_{i\in[n]}\tau_{1i}\mathbf{x}_{1i}\mathbf{x}_{1i}^T-\mathbb{E}\left[\tau_{ji}\mathbf{x}_{ji}\mathbf{x}_{ji}^T\right]\right|\right|_\infty+\left|\left|\frac{1}{mn}\sum_{j}\sum_{i}\tau_{ji}\mathbf{x}_{ji}\mathbf{x}_{ji}^T-\mathbb{E}\left[\tau_{ji}\mathbf{x}_{ji}\mathbf{x}_{ji}^T\right]\right|\right|_\infty\right) \\
&\quad\cdot||\widehat{\mathbf{w}}_t-\mathbf{w}^*||_1\ .
\end{aligned}$$

Using Hoeffding's inequality together with a union bound, we have with probability at least $1-\delta$,

$$\left|\left|\frac{1}{n}\sum_{i\in[n]}\tau_{1i}\mathbf{x}_{1i}\mathbf{x}_{1i}^T-\mathbb{E}\left[\tau_{ji}\mathbf{x}_{ji}\mathbf{x}_{ji}^T\right]\right|\right|_\infty\le L\left(\max_{j,i}||\mathbf{x}_{ji}||_\infty^2\right)\sqrt{\frac{2\log(2p/\delta)}{n}},$$

and

$$\left|\left|\frac{1}{mn}\sum_{j}\sum_{i}\tau_{ji}\mathbf{x}_{ji}\mathbf{x}_{ji}^T-\mathbb{E}\left[\tau_{ji}\mathbf{x}_{ji}\mathbf{x}_{ji}^T\right]\right|\right|_\infty\le L\left(\max_{j,i}||\mathbf{x}_{ji}||_\infty^2\right)\sqrt{\frac{2\log(2p/\delta)}{mn}}.$$

Combining the bounds, the proof of the lemma is complete. □

2.6.2 Proof of Lemma 9

Proof. The proof uses ideas presented in [Negahban et al., 2012]. By triangle inequality we have

$$\begin{aligned}||\widehat{\mathbf{w}}_{t+1}||_1 - ||\mathbf{w}^*||_1 =& ||\mathbf{w}^* + (\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_{S^c} + (\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1 - ||\mathbf{w}^*||_1 \\ \geq& ||\mathbf{w}^* + (\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_{S^c}||_1 - ||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1 - ||\mathbf{w}^*||_1 \\ =& ||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_{S^c}||_1 - ||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1.\end{aligned}$$

By the optimality of $\widehat{\mathbf{w}}_{t+1}$ for (2.4), we have

$$\widetilde{\mathcal{L}}_1(\widehat{\mathbf{w}}_{t+1}, \widehat{\mathbf{w}}_t) + \lambda_{t+1}||\widehat{\mathbf{w}}_{t+1}||_1 - \widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t) - \lambda_{t+1}||\mathbf{w}^*||_1 \leq 0.$$

Thus

$$\widetilde{\mathcal{L}}_1(\widehat{\mathbf{w}}_{t+1}, \widehat{\mathbf{w}}_t) - \widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t) + \lambda_{t+1}(||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_{S^c}||_1 - ||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1) \leq 0.$$

By the convexity of $\widetilde{\mathcal{L}}_1(\cdot, \widehat{\mathbf{w}}_t)$, we further have

$$\widetilde{\mathcal{L}}_1(\widehat{\mathbf{w}}_{t+1}, \widehat{\mathbf{w}}_t) - \widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t) \geq \langle \nabla\widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t), \widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*\rangle.$$

Thus by Hölder's inequality

$$\begin{aligned}0 \geq& \langle \nabla\widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t), \widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*\rangle + \lambda_{t+1}(||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_{S^c}||_1 - ||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1) \\ \geq& -||\nabla\widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t)||_\infty ||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 + \lambda_{t+1}(||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_{S^c}||_1 - ||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1).\end{aligned}$$

Under the assumption on λ_{t+1} we further have

$$\begin{aligned}0 &\geq -\frac{\lambda_{t+1}}{2}||\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*||_1+\lambda_{t+1}(||(\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*)_{S^c}||_1-||(\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*)_S||_1)\\ &= \frac{\lambda_{t+1}}{2}||(\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*)_{S^c}||_1-\frac{3\lambda_{t+1}}{2}||(\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*)_S||_1,\end{aligned}$$

which completes the proof. □

2.6.3 Proof of Theorem 6

Proof. For the term $\widetilde{\mathcal{L}}_1(\widehat{\mathbf{w}}_{t+1},\widehat{\mathbf{w}}_t)-\widetilde{\mathcal{L}}_1(\mathbf{w}^*,\widehat{\mathbf{w}}_t)$ we have

$$\begin{aligned}\widetilde{\mathcal{L}}_1(\widehat{\mathbf{w}}_{t+1},\widehat{\mathbf{w}}_t)-\widetilde{\mathcal{L}}_1(\mathbf{w}^*,\widehat{\mathbf{w}}_t) =&\mathcal{L}_1(\widehat{\mathbf{w}}_{t+1})+\left\langle\frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t)-\nabla\mathcal{L}_1(\widehat{\mathbf{w}}_t),\widehat{\mathbf{w}}_{t+1}\right\rangle\\ &-\mathcal{L}_1(\mathbf{w}^*)-\left\langle\frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t)-\nabla\mathcal{L}_1(\widehat{\mathbf{w}}_t),\mathbf{w}^*\right\rangle\\ \geq&\langle\nabla\mathcal{L}_1(\mathbf{w}^*),\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*\rangle+\kappa||\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*||_2^2\\ &+\left\langle\frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t)-\nabla\mathcal{L}_1(\widehat{\mathbf{w}}_t),\widehat{\mathbf{w}}_{t+1}\right\rangle\\ &-\left\langle\frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t)-\nabla\mathcal{L}_1(\widehat{\mathbf{w}}_t),\mathbf{w}^*\right\rangle\\ =&\left\langle\nabla\mathcal{L}_1(\mathbf{w}^*)+\frac{1}{m}\sum_{j\in[m]}\nabla\mathcal{L}_j(\widehat{\mathbf{w}}_t)-\nabla\mathcal{L}_1(\widehat{\mathbf{w}}_t),\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*\right\rangle\\ &+\kappa||\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*||_2^2\\ =&\langle\nabla\widetilde{\mathcal{L}}_1(\mathbf{w}^*,\widehat{\mathbf{w}}_t),\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*\rangle+\kappa||\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*||_2^2,\end{aligned}$$

where the first inequality we use the restricted strong convexity condition (5). Also by the optimality of $\widehat{\mathbf{w}}_{t+1}$ for (2.4), we have

$$\widetilde{\mathcal{L}}_1(\widehat{\mathbf{w}}_{t+1}, \widehat{\mathbf{w}}_t) - \widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t) + \lambda_{t+1}||\widehat{\mathbf{w}}_{t+1}||_1 - \lambda_{t+1}||\mathbf{w}^*||_1 \leq 0.$$

Combining above two inequalities we obtain with probability at least $1-\delta$:

$$\begin{aligned}
\lambda_{t+1}||\mathbf{w}^*||_1 - \lambda_{t+1}||\widehat{\mathbf{w}}_{t+1}||_1 \geq & \langle \nabla\widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t), \widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*\rangle + \kappa||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2^2 \\
\geq & - ||\nabla\widetilde{\mathcal{L}}_1(\mathbf{w}^*, \widehat{\mathbf{w}}_t)||_\infty ||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 + \kappa||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2^2 \\
\geq & - \frac{\lambda_{t+1}}{2}||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 + \kappa||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2^2.
\end{aligned}$$

By triangle inequality that $\lambda_{t+1}||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 \geq \lambda_{t+1}||\mathbf{w}^*||_1 - \lambda_{t+1}||\widehat{\mathbf{w}}_{t+1}||_1$, we have

$$\begin{aligned}
\kappa||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2^2 \leq & \frac{3\lambda_{t+1}}{2}||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 \\
= & \frac{3\lambda_{t+1}}{2}(||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1 + ||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_{S^c}||_1) \\
\leq & \frac{3\lambda_{t+1}}{2}(||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1 + 3||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1) \\
= & 6\lambda_{t+1}||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1 \\
\leq & 6\sqrt{s}\lambda_{t+1}||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_2 \\
\leq & 6\sqrt{s}\lambda_{t+1}||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2.
\end{aligned}$$

We get

$$||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2 \leq \frac{6\sqrt{s}\lambda_{t+1}}{\kappa}.$$

Substitute λ_{t+1} in (2.9) concludes the proof for ℓ_2 estimation error bound. For $||\widehat{\mathbf{w}}_{t+1}-\mathbf{w}^*||_1$, we know

$$\begin{aligned}||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 \leq & ||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1 + ||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_{S^c}||_1 \\ \leq & 4||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_1 \leq 4\sqrt{s}||(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*)_S||_2 \\ \leq & 4\sqrt{s}||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2 \leq \frac{24 s \lambda_{t+1}}{\kappa},\end{aligned}$$

which obtains the desired bound. □

2.6.4 Proof of Theorem 3

Proof. Theorem 3 follows from Theorem 6 after we verify some conditions. First, it is easy to see that the quadratic loss $L = 1, M = 0$. Under conditions of Theorem, with probability $1-\delta$,

$$\left|\left|\frac{1}{m}\sum_{j\in[m]} \nabla \mathcal{L}_j(\mathbf{w}^*)\right|\right|_\infty \lesssim \sigma\sigma_X \sqrt{\frac{\log(p/\delta)}{mn}}.$$

This follows from Corollary 5.17 of Vershynin [2010]. Furthermore, with probability at least $1-\delta$, we have

$$\max_{j\in[m], i\in[n]} ||\mathbf{x}_{ji}||_\infty \lesssim \sigma_X \sqrt{\log(mnp/\delta)}.$$

Finally,

$$||\widehat{\mathbf{w}}_0 - \mathbf{w}^*||_1 \lesssim \frac{s\sigma\sigma_X}{\kappa}\sqrt{\frac{\log(np/\delta)}{n}},$$

with probability at least $1-\delta$ [Wainwright, 2009, Meinshausen and Yu, 2009, Bickel et al., 2009]. Plugging these bounds into Theorem 6 completes the proof.

□

2.6.5 Proof of Corollary 7

Proof. The proof proceeds by recursively applying Theorem 6 and sum a geometric sequence. For notation simplicity let

$$\begin{aligned} a =& \frac{48s}{\kappa} \left|\left| \frac{1}{m} \sum_{j \in [m]} \nabla \mathcal{L}_j(\mathbf{w}^*) \right|\right|_\infty, \\ b =& \left(\frac{48sL}{\kappa} \left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^2 \right) \sqrt{\frac{4\log(2p/\delta)}{n}} \right), \\ c =& \frac{48sM}{\kappa} \left(\max_{j,i} ||\mathbf{x}_{ji}||_\infty^3 \right). \end{aligned}$$

By Theorem 6 we have

$$\begin{aligned} ||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_1 \leq& a + b||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1 + c||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1^2 \\ \leq& a + 2b||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1 \\ \leq& a + 2b(a + 2b||\widehat{\mathbf{w}}_{t-1} - \mathbf{w}^*||_1) \leq \dots \\ \leq& a \sum_{k=0}^{t} (2b)^k + (2b)^{t+1}||\widehat{\mathbf{w}}_0 - \mathbf{w}^*||_1. \\ =& \frac{a(1-(2b)^{t+1})}{1-2b} + (2b)^{t+1}||\widehat{\mathbf{w}}_0 - \mathbf{w}^*||_1, \end{aligned} \tag{2.16}$$

which completes the ℓ_1 estimation error bound. For $||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2$, we first use (2.16) to obtain

$$||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1 \leq \frac{a(1-(2b)^t)}{1-(2b)} + (2b)^t||\widehat{\mathbf{w}}_0 - \mathbf{w}^*||_1.$$

Then apply Theorem 6 to obtain that

$$
\begin{aligned}
||\widehat{\mathbf{w}}_{t+1} - \mathbf{w}^*||_2 \leq & \frac{a}{4\sqrt{s}} + \frac{(2b)}{4\sqrt{s}}||\widehat{\mathbf{w}}_t - \mathbf{w}^*||_1 \leq \frac{a}{4\sqrt{s}} + \frac{b}{4\sqrt{s}}\left(\frac{a(1-(2b)^t)}{1-(2b)} + (2b)^t||\widehat{\mathbf{w}}_0 - \mathbf{w}^*||_1\right) \\
= & \frac{1}{4\sqrt{s}}\left(a + \frac{a((2b)-(2b)^{t+1})}{1-(2b)}\right) + \frac{(2b)^{t+1}||\widehat{\mathbf{w}}_0 - \mathbf{w}^*||_1}{4\sqrt{s}} \\
= & \frac{a(1-(2b)^{t+1})}{4\sqrt{s}(1-(2b))} + \frac{(2b)^{t+1}||\widehat{\mathbf{w}}_0 - \mathbf{w}^*||_1}{4\sqrt{s}},
\end{aligned}
$$

which concludes the proof. □

2.6.6 Proof of Lemma 10

Proof. By the definition of $\mathcal{L}_j(\mathbf{w})$, we have

$$
\frac{1}{m}\sum_{j\in[m]} \nabla\mathcal{L}_j(\mathbf{w}^*) = \frac{1}{mn}\sum_{j\in[m]}\sum_{i\in[n]} \mathbf{x}_{ji}\left(y_{ji} - \frac{y_{ji}}{1+\exp(-y_{ji}\langle \mathbf{w}, \mathbf{x}_{ji}\rangle)}\right).
$$

It is easy to check that

$$
\mathbb{E}\left[y_{ji} - \frac{y_{ji}}{1+\exp(-y_{ji}\langle \mathbf{w}, \mathbf{x}_{ji}\rangle)}\right] = 0, \quad \text{and} \quad \left|y_{ji} - \frac{y_{ji}}{1+\exp(- - y_{ji}\langle \mathbf{w}, \mathbf{x}_{ji}\rangle)}\right| \leq 1
$$

and thus

$$
\begin{aligned}
&\mathbb{E}\left[\mathbf{x}_{ji}\left(y_{ji} - \frac{y_{ji}}{1+\exp(-y_{ji}\langle \mathbf{w}, \mathbf{x}_{ji}\rangle)}\right)\right] = 0, \\
&\left|\left|\mathbf{x}_{ji}\left(y_{ji} - \frac{y_{ji}}{1+\exp(- - y_{ji}\langle \mathbf{w}, \mathbf{x}_{ji}\rangle)}\right)\right|\right|_\infty \leq \max_{ji}\left(||\mathbf{x}_{ji}||_\infty\right).
\end{aligned}
$$

Appling Azuma-Hoeffding inequality [Hoeffding, 1963] and the union bound over $[p]$ leads to the desired bound. □

CHAPTER 3

MEMORY AND COMMUNICATION EFCIENT DISTRIBUTED STOCHASTIC OPTIMIZATION WITH MINIBATCH-PROX

3.1 Motivation and problem set-up

Consider the stochastic convex optimization (generalized learning) problem [Nemirovskii et al., 1983, Vapnik, 1995, Shalev-Shwartz et al., 2009]:

$$\min_{\mathbf{w}\in\Omega} \phi(\mathbf{w}) := \mathbb{E}_{\xi\sim D}\left[\ell(\mathbf{w},\xi)\right] \tag{3.1}$$

where our goal is to learn a predictor $\mathbf{w}$ from the convex domain Ω given the convex instantaneous (loss) function $\ell(\mathbf{w},\xi)$ and i.i.d. samples $\xi_1, \xi_2, \ldots$ from some unknown data distribution D. When optimizing on a single machine, stochastic approximation methods such as stochastic gradient descent (SGD) or more generally stochastic mirror descent, are ideally suited for the problem as they typically have optimal sample complexity requirements, and run in linear time in the number of samples, and thus also have optimal runtime. Focusing on an ℓ_2 bounded domain with $B = \sup_{\mathbf{w}\in\Omega} ||\mathbf{w}||$ and L-Lipschitz loss, the min-max optimal sample complexity is $n(\varepsilon) = \mathcal{O}(L^2B^2/\varepsilon^2)$, and this is achieved by SGD using $\mathcal{O}(n(\epsilon))$ vector operations. Furthermore, if examples are obtained one at a time (in a streaming setting or through access to a "button" generating examples), we only need to store $\mathcal{O}(1)$ vectors in memory.

The situation is more complex in the distributed setting where no single method is known that is optimal with respect to sample complexity, runtime, memory *and* communication. Specifically, consider m machines where each machine $i = 1, ..., m$ receives samples $\xi_{i1}, \xi_{i2}, ...$ drawn from the same distribution D. This can equivalently be thought of as randomly distributing samples across m servers. We also assume the objective is β-smooth, taking $L, \beta = \mathcal{O}(1)$ in our presentation of results. The goal is to find a predictor $\widehat{\mathbf{w}} \in \Omega$ satisfying $\mathbb{E}\left[\phi(\widehat{\mathbf{w}}) - \min_{\mathbf{w}\in\Omega} \phi(\mathbf{w})\right] \leq \varepsilon$ using *the smallest possible number of samples* per machine, *the minimal elapsed runtime*, and *the smallest amount of communication*, and also *minimal memory* on each machine (again, when examples are received or generated one at a time). Ideally, we could hope for a method with linear speedup, i.e. $\mathcal{O}(n(\epsilon)/m)$ runtime, using the statistically optimal number of samples $\mathcal{O}(n(\epsilon))$ and constant or near-constant communication and memory. Throughout we measure runtime in terms of vector operations, memory in terms of number of vectors that need to be stored on each machine and communication in terms of number of vectors sent per machine[1]. These resource requirements are summarized in Table 7.1.

One simple approach for distributed stochastic optimization is minibatch SGD [Cotter et al., 2011, Dekel et al., 2012], where in each update we use a gradient estimate based on mb examples: b examples from each of the m machines. Distributed minibatch SGD attains optimal statistical performance with $\mathcal{O}\left(n(\varepsilon)/m\right)$ runtime, as long as the minibatch size is not too large: Dekel et al. [2012] showed that the minibatch size can be as large as $bm = \mathcal{O}(\sqrt{n(\varepsilon)})$, and Cotter et al. [2011] showed that with acceleration this can be increased to $bm = \mathcal{O}(n(\varepsilon)^{3/4})$. Using this maximal minibatch size for accelerated minibatch SGD thus yields a statistically optimal method with linear speedup in runtime, $\mathcal{O}(1)$ memory usage, and $\mathcal{O}(n(\varepsilon)^{1/4})$ rounds of communication–see Table 7.1. This is the most communication-efficient method with true linear speedup we are aware of.

An alternative approach is to use distributed optimization to optimize the regularized empirical objective:

$$\min_{\mathbf{w}} \; \phi_S(\mathbf{w}) + \frac{\nu}{2}||\mathbf{w}||^2, \tag{3.2}$$

1. In all methods involved, communication is used to average vectors across machines and make the result known to one or all machines. We are actually counting the number of such operations.

where ϕ_S is the empirical objective on $n(\epsilon)$ i.i.d. samples, distributed across the machines and $\nu = \mathcal{O}(L/(B\sqrt{n(\varepsilon)}))$. A naive approach here is to use accelerate gradient descent, distributing the gradient computations, but this, as well as approaches based on ADMM [Boyd et al., 2011], are dominated by minibatch SGD (Shamir and Srebro 2014 and see also Table 7.1). Better alternatives take advantage of the stochastic nature of the problem: DANE [Shamir et al., 2014] requires only $\mathcal{O}(B^2m)$ rounds of communication for squared loss problems, while DiSCO [Zhang and Xiao, 2015] and AIDE [Reddi et al., 2016]) reduce this further to $\mathcal{O}(B^{1/2}m^{1/4})$ rounds of communication. However, these communication-efficient methods usually require expensive computation on each local machine, solving an optimization problem on all local data at each iteration. Even if this can be done in near-linear time, it is still difficult to obtain computational speedup compared with single machine solution, and certainly not linear speedups—see Table 7.1. Furthermore, since each round of these methods involves optimization over a fixed training set, this training set must be stored thus requiring $n(\varepsilon)/m$ memory per machine.

Designing stochastic distributed optimization problems with linear, or near-linear, speedups, and low communication and memory requirements is thus still an open problem. We make progress in this chapter analyzing and presenting methods with near-linear speedups and better communication and memory requirements. As with the analysis of DANE, DiSCO and AIDE, our analysis is rigorous only for least squared problems, and so all results should be taken in that context (the methods themselves are applicable to any distributed stochastic convex optimization problem).

Our contributions

- We first apply the recently proposed distributed SVRG (DSVRG) algorithm for regularized loss minimization to the distributed stochastic convex optimization problem, and show that on least square problems it can achieve near-linear speedup with very

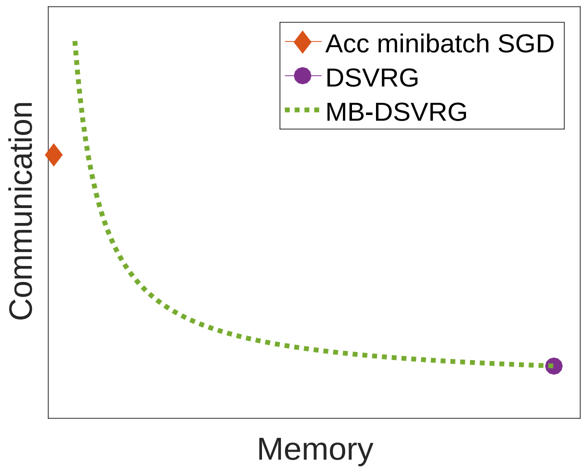


Figure 3.1: Trade-offs between memory and communication for the proposed MP-DSVRG approach.

low communication, but with high memory cost—see DSVRG in Table 7.1.

- We propose a novel algorithm that improves the memory cost, which we call minibatch-prox with DSVRG (MP-DSVRG). For least square problems it achieves near-linear speedup with communication cost that is higher than DSVRG but increases only logarithmically with $n(\varepsilon)$, but with much lower memory requirements. Moreover, our algorithm is flexible, allowing to trade off between communication and memory (depicted in Figure 3.1), without affecting the computational efficiency. Our method is based on careful combinations of inexact minibatch proximal update, communication-efficient optimization and linearly convergent stochastic gradient algorithms for finite-sums.

- As indicated above, our method is based on minibatch proximal update. That is, a minibatch approach where in each iteration a non-linearized problem is solved on a stochastic minibatch. This can be viewed as a minibatch generalization to the passive-aggressive algorithm [Crammer et al., 2006] and has been considered in various contexts [Kulis and Bartlett, 2010, Toulis and Airoldi, 2014]. We show that such an approach achieves the optimal statistical rate in terms of the number of samples used

	Samples	Communication	Computation	Memory
Ideal Solution	$n(\varepsilon)$	$\mathcal{O}(1)$	$n(\varepsilon)/m$	$\mathcal{O}(1)$
Accelerated GD	$n(\varepsilon)$	$B^{1/2}n(\varepsilon)^{1/4}$	$B^{1/2}n(\varepsilon)^{5/4}/m$	$n(\varepsilon)/m$
Acc. Minibatch SGD	$n(\varepsilon)$	$B^{1/2}n(\varepsilon)^{1/4}$	$n(\varepsilon)/m$	$\mathcal{O}(1)$
DANE	$n(\varepsilon)$	B^2m	$B^2n(\varepsilon)$	$n(\varepsilon)/m$
DiSCO	$n(\varepsilon)$	$B^{1/2}m^{1/4}$	$B^{1/2}n(\varepsilon)/m^{3/4}$	$n(\varepsilon)/m$
AIDE	$n(\varepsilon)$	$B^{1/2}m^{1/4}$	$B^{1/2}n(\varepsilon)/m^{3/4}$	$n(\varepsilon)/m$
DSVRG	$n(\varepsilon)$	$\mathcal{O}(1)$	$n(\varepsilon)/m$	$n(\varepsilon)/m$
MP-DSVRG ($b \leq b_{\max}$)	$n(\varepsilon)$	$n(\varepsilon)/(mb)$	$n(\varepsilon)/m$	b
MP-DSVRG ($b = b_{\max}$)	$n(\varepsilon)$	$\mathcal{O}(1)$	$n(\varepsilon)/m$	$n(\varepsilon)/m$

Table 3.1: Summary of resources required by different approaches to distributed stochastic least squares problems, in units of vector operations/communications/memory per machine, ignoring constants and log-factors, here $b_{\max} = n(\varepsilon)/m$.

independent of the number of iterations, i.e. with *any* minibatch size. This significantly improves over the previous analysis of Li et al., as the guarantee is better, it entirely avoid the dependence on the minibatch size and does not rely on additional assumptions as in Li et al.. The guarantee holds for any Lipschitz (even non-smooth) objective. Furthermore, to make the minibatch proximal iterate more practical and useful in distributed setting, we also extend the analysis to algorithms which solve each minibatch subproblem inexactly. Our analysis of exact and inexact minibatch proximal updates may be of independent interest and useful in other contexts and as a basis for other methods.

Notations We denote by $\mathbf{w}_* = \operatorname{argmin}_{\mathbf{w}\in\Omega} \ \phi(\mathbf{w})$ the optimal solution to (3.1). Throughout the chapter, we assume the instantaneous function $\ell(\mathbf{w}, \xi)$ is L-Lipschitz and λ-strongly convex in $\mathbf{w}$ for some $\lambda \geq 0$ on the domain Ω:

$$\left|\ell(\mathbf{w}, \xi) - \ell(\mathbf{w}', \xi)\right| \leq L||\mathbf{w} - \mathbf{w}'||,$$
$$\ell(\mathbf{w}, \xi) - \ell(\mathbf{w}', \xi) \geq \left\langle \nabla\ell(\mathbf{w}', \xi),\, \mathbf{w} - \mathbf{w}' \right\rangle + \frac{\lambda}{2}||\mathbf{w} - \mathbf{w}'||^2, \qquad \forall \mathbf{w}, \mathbf{w}' \in \Omega.$$

Sometimes we also assume $\ell(\mathbf{w}, \xi)$ is β-smooth in $\mathbf{w}$:

$$\ell(\mathbf{w}, \xi) - \ell(\mathbf{w}', \xi) \leq \left\langle \nabla \ell(\mathbf{w}', \xi),\, \mathbf{w} - \mathbf{w}' \right\rangle + \frac{\beta}{2} ||\mathbf{w} - \mathbf{w}'||^2, \qquad \forall \mathbf{w}, \mathbf{w}' \in \Omega.$$

For distributed stochastic optimization, our analysis focuses on the least squares loss $\ell(\mathbf{w}, \xi) = \frac{1}{2}(\mathbf{w}^\top \mathbf{x} - y)^2$ where $\xi = (\mathbf{x}, y)$.

3.2 Distributed SVRG for stochastic convex optimization

Recently, Lee et al. [2017a] suggested using fast randomized optimization algorithms for finite-sums, and in particular the SVRG algorithm, as a distributed optimization approach for (3.2). The authors noted that, for SVRG, when the the sample size $n(\varepsilon)$ dominates the problem's condition number β/ν where β is the smoothness parameter of $\ell(\mathbf{w}, \xi)$, the time complexity is dominated by computing the batch gradients. This operation can be trivially parallelized. The stochastic updates, on the other hand, can be implemented on a single machine while the other machines wait, with the only caveat being that only sampling-without-replacement can be implemented this way. The use of without-replacement sampling was theoretically justified in a recent analysis by Shamir [2016].

In the distributed stochastic convex optimization setting considered here, DSVRG in fact achieves linear speedup in certain regime as follows. In each iteration of the algorithm, each machine first computes its local gradient and average them with one communication round to obtain the global batch gradient, and then a *single* machine performs the SVRG stochastic updates by processing its local data once (sampling the $n(\varepsilon)/m$ examples without replacement). By the linear convergence of SVRG, as long as the number of stochastic updates $n(\varepsilon)/m$ is larger than $\beta/\nu = \mathcal{O}(\beta B \sqrt{n(\varepsilon)}/L)$, the algorithm converges to $\mathcal{O}(\epsilon)$-suboptimality (in both the empirical and stochastic objective) in $\mathcal{O}(\log 1/\varepsilon) = \mathcal{O}\left(\log n(\varepsilon)\right)$

iterations; and this condition is satisfied[2] for $n(\varepsilon) \gtrsim m^2$.

Clearly, in the above regime, each iteration of DSVRG uses two rounds of communications and the total communication complexity is $\mathcal{O}\left(n(\varepsilon)\right)$. On the other hand, the computation for each machine is compute the local gradient (in time $\mathcal{O}(n(\varepsilon)/m)$) in each iteration, resulting in a total time complexity of $\mathcal{O}(n(\varepsilon)\log n(\varepsilon)/m)$. This explains the DSVRG entry in Table 7.1.

Being communication- and computation-efficient, DSVRG requires each machine to store a portion of the sample set for ERM to make multiple passes over them, and is therefore not memory-efficient. In fact, this disadvantage is shared by previously known communication-efficient distributed optimization algorithms, including DANE, DiSCO, and AIDE. In order to develop a memory- and communication-efficient algorithm for distributed stochastic optimization, we need to bypass the ERM setting and this is enabled by the following minibatch-prox algorithm.

3.3 The minibatch-prox algorithm for stochastic optimization

In this section, we describe and analyze the minibatch-prox algorithm for stochastic optimization, which allows us to use arbitrarily large minibatch size without slowing down the convergence rate. We first present the basic version where each proximal objective is solved exactly for each minibatch, which achieves the optimal convergence rate. Then, we show that if each minibatch objective is solved accurately enough, the algorithm still converges at the optimal rate, opening the opportunity for efficient implementations.

2. If $n(\varepsilon) \gtrsim m^2$ does not hold, we can use a "hot-potato" style algorithm where we process all data once on machine i and pass the predictor to machine $i+1$ until we obtain sufficiently many stochastic updates. But then the computation efficiency deteriorates and we no longer have linear speedup in runtime.

3.3.1 Exact minibatch-prox

The "exact" minibatch-prox is defined by the following iterates: for $t = 1, \dots,$

$$\mathbf{w}_t = \operatorname*{argmin}_{\mathbf{w}\in\Omega} \; f_t(\mathbf{w}),$$
$$\text{where} \quad f_t(\mathbf{w}) := \phi_{I_t}(\mathbf{w}) + \frac{\gamma_t}{2}\|\mathbf{w} - \mathbf{w}_{t-1}\|^2 = \frac{1}{b}\sum_{\xi\in I_t} \ell(\mathbf{w}, \xi) + \frac{\gamma_t}{2}\|\mathbf{w} - \mathbf{w}_{t-1}\|^2, \qquad (3.3)$$

$\gamma_t > 0$ is the (inverse) stepsize parameter at time t, and I_t is a set of b samples from the unknown distribution D. To understand the updates in (3.3), we first observe by the first order optimality condition for $f_t(\mathbf{w})$ that

$$\nabla\phi_{I_t}(\mathbf{w}_t) + \gamma_t(\mathbf{w}_t - \mathbf{w}_{t-1}) \in -\mathcal{N}_\Omega(\mathbf{w}_t), \qquad (3.4)$$

where $\nabla\phi_{I_t}(\mathbf{w}_t)$ is some subgradient of $\phi_{I_t}(\mathbf{w})$ at $\mathbf{w}_t$, and $\mathcal{N}_\Omega(\mathbf{w}_t) = \{\mathbf{y} | \, \langle \mathbf{w} - \mathbf{w}_t, \, \mathbf{y}\rangle \le 0, \, \forall \mathbf{w} \in \Omega\}$ is the normal cone of Ω at $\mathbf{w}_t$. Equivalently, the above condition implies

$$\mathbf{w}_t = P_\Omega\left(\mathbf{w}_{t-1} - \frac{1}{\gamma_t}\nabla\phi_{I_t}(\mathbf{w}_t)\right), \qquad (3.5)$$

where $P_\Omega(\mathbf{w})$ denotes the projection of $\mathbf{w}$ onto Ω. The update rule (3.5) resembles that of the standard minibatch gradient descent, except the gradient is evaluated at the "future" iterate.

Proximal steps, of the form (3.3) or equivalently (3.5), are trickier to implement compared to (stochastic) gradient steps, as they involve optimization of a subproblem, instead of merely computing and adding gradients. Nevertheless, they have been suggested, used and studied in several contexts. Crammer et al. [2006] proposed the "passive aggressive" update rule, where a margin-based loss from a single example with a quadratic penalty is minimized—this corresponds to (3.3) with a "batch size" of one. More general loss functions, still for "batch sizes" of one, were also analyzed in the online learning setting [Cheng et al., 2007, Kulis and Bartlett, 2010]. For finite-sum objectives, methods based on incremental/stochastic proximal updates were studied by Bertsekas [2011, 2015], Defazio [2016]. Needell and Tropp [2014] analyzed a randomized block Kaczmarz method in the context of solving linear systems, which also minimizes the empirical loss on a randomly sampled minibatch. To the best of our knowledge, no prior work has analyzed the general minibatch variant of proximal updates for stochastic optimization except Li et al.. However, the analysis of Li et al. assumes a stringent condition which is hard to verify (and is often violated) in practice, which we will discuss in detail in this section.

The following lemma provides the basic property of the update at each iteration.

Lemma 13. *For any* $\mathbf{w} \in \Omega$, *we have*

$$\frac{\lambda + \gamma_t}{\gamma_t} ||\mathbf{w}_t - \mathbf{w}||^2 \le ||\mathbf{w}_{t-1} - \mathbf{w}||^2 - ||\mathbf{w}_{t-1} - \mathbf{w}_t||^2 - \frac{2}{\gamma_t} \left(\phi_{I_t}(\mathbf{w}_t) - \phi_{I_t}(\mathbf{w}) \right). \tag{3.6}$$

To derive the convergence guarantee, we need to relate $\phi_{I_t}(\mathbf{w}_t)$ to $\phi(\mathbf{w})$. The analysis of Li et al. for minibatch-prox made the assumption that for all $t \ge 1$:

$$\mathbb{E}_{I_t} \left[D_{\phi}(\mathbf{w}_t; \mathbf{w}_{t-1}) \right] \le \mathbb{E}_{I_t} \left[D_{\phi_{I_t}}(\mathbf{w}_t; \mathbf{w}_{t-1}) \right] + \frac{\gamma_t}{2} ||\mathbf{w}_t - \mathbf{w}_{t-1}||^2, \tag{3.7}$$

where $D_f(\mathbf{w}, \mathbf{w}') = f(\mathbf{w}) - f(\mathbf{w}') - \left\langle \nabla f(\mathbf{w}'), \mathbf{w} - \mathbf{w}' \right\rangle$ denotes the Bregman divergence defined by the potential function f. This condition is hard to verify, and may constrain the stepsize to be very small. For example, as the authors argued, if $\ell(\mathbf{w}, \xi)$ is β-smooth with respect to $\mathbf{w}$, we have

$$D_{\phi}(\mathbf{w}_t; \mathbf{w}_{t-1}) \le \frac{\beta}{2} ||\mathbf{w}_t - \mathbf{w}_{t-1}||^2,$$

and combined with the fact that $D_{\phi_{I_t}}(\mathbf{w}_t; \mathbf{w}_{t-1}) \ge 0$, one can guarantee (3.7) by setting

$\gamma_t \geq \beta$. However, to obtain the optimal convergence rate, Li et al. needed to set $\gamma_t = \mathcal{O}(\sqrt{T/b})$ which would imply $b = \mathcal{O}(T)$ in order to have $\gamma_t \geq \beta$. In view of this implicit constraint that the minibatch size b can not be too large, the analysis of Li et al. does not really show advantage of minibatch-prox over minibatch SGD, whose optimal minibatch size is precisely $b = \mathcal{O}(T)$.

Our analysis is free of any additional assumptions. The key observation is that, when b is large, we expect $\phi_{I_t}(\mathbf{w})$ to be close to $\phi(\mathbf{w})$. Define the stochastic objective

$$F_t(\mathbf{w}) := \mathbb{E}_{I_t}\left[f_t(\mathbf{w})\right] = \phi(\mathbf{w}) + \frac{\gamma_t}{2}||\mathbf{w} - \mathbf{w}_{t-1}||^2. \tag{3.8}$$

Then $\mathbf{w}_t$ is the "empirical risk minimizer" of $F_t(\mathbf{w})$ as it solves the empirical version $f_t(\mathbf{w})$ with b samples. Using a stability argument [Shalev-Shwartz et al., 2009], we can establish the "generalization" performance for the (inexact) minimizer of the minibatch objective.

Lemma 14. *For the minibatch-prox algorithm,we have*

$$\left|\mathbb{E}_{I_t}\left[\phi(\mathbf{w}_t) - \phi_{I_t}(\mathbf{w}_t)\right]\right| \leq \frac{4L^2}{(\lambda + \gamma_t)b}.$$

Moreover, if a possibly randomized algorithm $\mathcal{A}$ minimizes $f_t(\mathbf{w})$ up to an error of η_t, i.e., $\mathcal{A}$ returns an approximate solution $\widetilde{\mathbf{w}}_t$ such that $\mathbb{E}_{\mathcal{A}}\left[f_t(\widetilde{\mathbf{w}}_t) - f_t(\mathbf{w}_t)\right] \leq \eta_t$, we have

$$\left|\mathbb{E}_{I_t,\mathcal{A}}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi_{I_t}(\mathbf{w}_t)\right]\right| \leq \frac{4L^2}{(\lambda + \gamma_t)b} + \sqrt{\frac{2L^2\eta_t}{\lambda + \gamma_t}}.$$

Combining Lemma 13 and Lemma 14, we obtain the following key lemma regarding the progress on the stochastic objective at each iteration of minibatch-prox.

Lemma 15. *For iteration t of exact minibatch-prox, we have for any $\mathbf{w} \in \Omega$ that*

$$\frac{\lambda + \gamma_t}{\gamma_t}\mathbb{E}_{I_t}||\mathbf{w}_t - \mathbf{w}||^2 \leq ||\mathbf{w}_{t-1} - \mathbf{w}||^2 - \frac{2}{\gamma_t}\mathbb{E}_{I_t}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w})\right] + \frac{8L^2}{\gamma_t(\lambda + \gamma_t)b}. \tag{3.9}$$

We are now ready to bound the overall convergence rates of minibatch-prox.

Theorem 16 (Convergence of exact minibatch-prox — weakly convex $\ell(\mathbf{w}, \xi)$)**.** *For L-Lipschitz instantaneous function $\ell(\mathbf{w}, \xi)$, set $\gamma = \sqrt{\frac{8T}{b}} \cdot \frac{L}{||\mathbf{w}_0 - \mathbf{w}_*||}$ for $t = 1, \dots, T$ in minibatch-prox. Then for $\widehat{\mathbf{w}}_T = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$, we have*

$$\mathbb{E}\left[\phi(\widehat{\mathbf{w}}_T) - \phi(\mathbf{w}_*)\right] \leq \frac{\sqrt{8}L}{\sqrt{bT}} ||\mathbf{w}_0 - \mathbf{w}_*||.$$

Theorem 17 (Convergence of exact minibatch-prox — strongly convex $\ell(\mathbf{w}, \xi)$)**.** *For L-Lipschitz and λ-strongly convex instantaneous function $\ell(\mathbf{w}, \xi)$, set $\gamma_t = \frac{\lambda(t-1)}{2}$ for $t = 1, \dots, T$ in minibatch-prox. Then for $\widehat{\mathbf{w}}_T = \frac{2}{T(T+1)} \sum_{t=1}^{T} t\mathbf{w}_t$, we have*

$$\mathbb{E}\left[\phi(\widehat{\mathbf{w}}_T) - \phi(\mathbf{w}_*)\right] \leq \frac{16L^2}{\lambda b(T+1)}.$$

3.3.2 Inexact minibatch-prox

We now study the case where instead of solving the subproblems $f_t(\mathbf{w})$ exactly, we only solve it approximately to sufficient accuracy. The "inexact" minibatch-prox uses a possibly randomized algorithm $\mathcal{A}$ for approximately solving one subproblem on a minibatch in each iteration, and generates the following iterates: for $t = 1, \dots,$

$$\widetilde{\mathbf{w}}_t \approx \bar{\mathbf{w}}_t := \operatorname*{argmin}_{\mathbf{w} \in \Omega} \widetilde{f}_t(\mathbf{w}) \qquad \text{where} \quad \widetilde{f}_t(\mathbf{w}) := \phi_{I_t}(\mathbf{w}) + \frac{\gamma_t}{2} ||\mathbf{w} - \widetilde{\mathbf{w}}_{t-1}||^2, \tag{3.10}$$

$$\text{and} \qquad \mathbb{E}_{\mathcal{A}}\left[\widetilde{f}_t(\widetilde{\mathbf{w}}_t) - \widetilde{f}_t(\bar{\mathbf{w}}_t)\right] \leq \eta_t.$$

Analogous to Lemma 15, we can derive the following lemma using stability of inexact minimizers.

Lemma 18. *Fix any $\mathbf{w} \in \Omega$. For iteration t of inexact minibatch-prox, we have*

$$\mathbb{E}_{I_t,\mathcal{A}}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w})\right] \le \frac{\gamma_t}{2}\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_{t-1} - \mathbf{w}||^2 - \frac{\lambda + \gamma_t}{2}\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t - \mathbf{w}||^2 + \frac{4L^2}{(\lambda + \gamma_t)b}$$
$$+ \sqrt{\frac{2L^2\eta_t}{\lambda + \gamma_t}} + \sqrt{2(\lambda + \gamma_t)\eta_t} \cdot \sqrt{\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t - \mathbf{w}||^2}. \quad (3.11)$$

Note that when $\eta_t = 0$, the above guarantee reduces to that of exact minibatch-prox.

We now show that when the minibatch subproblems are solved sufficiently accurately, we still obtain the $\mathcal{O}(1/\sqrt{bT})$ rate for weakly-convex loss and $\mathcal{O}(1/(\lambda bT))$ rate for strongly-convex loss.

Theorem 19 (Convergence of inexact minibatch-prox — weakly convex $\ell(\mathbf{w}, \xi)$)**.** *For L-Lipschitz instantaneous function $\ell(\mathbf{w}, \xi)$, set $\gamma_t = \gamma = \sqrt{\frac{8T}{b}} \cdot \frac{L}{||\mathbf{w}_0 - \mathbf{w}_*||}$ for all $t \ge 1$ in inexact minibatch-prox. Assume that for all $t \ge 1$, the error in minimizing $\widetilde{f}_t(\mathbf{w})$ satisfies for some $\delta > 0$ that*

$$\mathbb{E}_{\mathcal{A}}\left[\widetilde{f}_t(\widetilde{\mathbf{w}}_t) - \min_{\mathbf{w}} \widetilde{f}_t(\mathbf{w})\right] \le \min\left(c_1\left(\frac{T}{b}\right)^{\frac{1}{2}}, c_2\left(\frac{T}{b}\right)^{\frac{3}{2}}\right) \cdot \frac{L||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||}{t^{2+2\delta}}.$$

Then for $\widehat{\mathbf{w}}_T = \frac{1}{T}\sum_{t=1}^{T}\widetilde{\mathbf{w}}_t$, we have $\mathbb{E}\left[\phi(\widehat{\mathbf{w}}_T) - \phi(\mathbf{w}_*)\right] \le \frac{c_3 L||\mathbf{w}_0 - \mathbf{w}_*||}{\sqrt{bT}}$, where c_3 only depends on c_1, c_2 and δ. For example, by setting $c_1 = 10^{-4}, c_2 = 10^{-4}, \delta = 1/2$, we have

$$\mathbb{E}\left[\phi(\widehat{\mathbf{w}}_T) - \phi(\mathbf{w}_*)\right] \le \frac{\sqrt{10}L||\mathbf{w}_0 - \mathbf{w}_*||}{\sqrt{bT}}.$$

Theorem 20 (Convergence of inexact minibatch-prox — strongly convex $\ell(\mathbf{w}, \xi)$)**.** *For L-Lipschitz and λ-strongly convex instantaneous function $\ell(\mathbf{w}, \xi)$, set $\gamma_t = \frac{\lambda(t-1)}{2}$ for $t = 1, \ldots$ in inexact minibatch-prox. Assume that for all $t \ge 1$, the error in minimizing $\widetilde{f}_t(\mathbf{w})$ satisfies* *for some $\delta > 0$ that*

$$\mathbb{E}_{\mathcal{A}}\left[\widetilde{f}_t(\widetilde{\mathbf{w}}_t) - \min_{\mathbf{w}} \widetilde{f}_t(\mathbf{w})\right] \le \min\left(c_1\left(\frac{T}{b}\right), c_2\left(\frac{T}{b}\right)^2\right) \cdot \frac{L^2}{t^{3+2\delta}\lambda}.$$

Then for $\widehat{\mathbf{w}}_T = \frac{2}{T(T+1)}\sum_{t=1}^{T} t\widetilde{\mathbf{w}}_t$, we have $\mathbb{E}\left[\phi(\widehat{\mathbf{w}}_T) - \phi(\mathbf{w}_*)\right] \le \frac{c_3 L^2}{\lambda b T}$, where c_3 only depends on c_1, c_2 and δ.

Remark 1. *The final inequalities in Theorem 16 and 19 actually apply more generally to all predictors in the domain. That is, our proofs still hold with $\mathbf{w}^*$ replaced by any $\mathbf{w} \in \Omega$:*

$$\mathbb{E}\left[\phi(\widehat{\mathbf{w}}_T) - \phi(\mathbf{w})\right] \le \mathcal{O}\left(\frac{L||\mathbf{w}_0 - \mathbf{w}||}{\sqrt{bT}}\right), \qquad \mathbf{w} \in \Omega.$$

This allows us to compete with any predictor in the domain (other than the minimizer). For example, in order to compete on $\phi(\mathbf{w})$ with the set of predictors with small norm $\{\mathbf{w} : ||\mathbf{w}|| \le B\}$, we can set the domain $\Omega = \mathbb{R}^d$ and initialize with $\mathbf{w}_0 = \mathbf{0}$. In view of the above inequality, we still obtain the optimal rate $\mathcal{O}\left(\frac{LB}{\sqrt{bT}}\right)$ from minibatch-prox by solving simpler, unconstrained subproblems (though we might have $||\widehat{\mathbf{w}}_T|| > B$).

3.4 Communication-efficient distributed minibatch-prox with DANE

As discussed in Section 3.5, it is also possible to use other efficient distributed optimization solver for minibatch-prox. Here we present a novel method that use the distributed optimization algorithm DANE [Shamir et al., 2014] and its accelerated variant AIDE [Reddi et al., 2016] for solving (3.16), which define better local objectives than EMSO and take into consideration the similarity between local objectives.

We detail our algorithm, named MP-DANE, in Algorithm 2. The algorithm consists of three nested loops, where t, r and k are iteration counters for minibatch-prox (the outer

Algorithm 2 MP-DANE for distributed stochastic convex optimization.

```
Initialize $\mathbf{w}_0$.
for $t = 1, 2, \ldots, T$ do
  Each machine $i$ draws a minibatch $I_t^{(i)}$ of $b$ samples from the underlying data distribution.
  Initialize $\mathbf{y}_0 \leftarrow \mathbf{w}_{t-1}, \quad \mathbf{x}_0 \leftarrow \mathbf{w}_{t-1}$.
  for $r = 1, 2, \ldots, R$ do
    Initialize $\mathbf{z}_0 \leftarrow \mathbf{y}_{r-1}$, $\alpha_0 = \sqrt{\gamma/(\gamma+\kappa)}$.
    for $k = 1, 2, \ldots, K$ do
      1. All machines perform one round of communication to compute the average gradient
         $$\nabla \phi_{I_t}(\mathbf{z}_{k-1}) \leftarrow \frac{1}{m} \sum_{i=1}^{m} \nabla \phi_{I_t^{(i)}}(\mathbf{z}_{k-1}).$$
      2. Each machine $i$ approximately solves the local objective to $\theta$-accuracy:
         $$\text{apply prox-SVRG to find } \mathbf{z}_k^{(i)} \quad \text{s.t.} \quad ||\mathbf{z}_k^{(i)} - \mathbf{z}_k^{(i)*}|| \le \theta ||\mathbf{z}_{k-1} - \mathbf{z}_k^{(i)*}||$$
         $$\text{where } \mathbf{z}_k^{(i)^*} = \operatorname*{argmin}_{\mathbf{z} \in \Omega} \ \phi_{I_t^{(i)}}(\mathbf{z}) + \left\langle \nabla \phi_{I_t}(\mathbf{z}_{k-1}) - \nabla \phi_{I_t^{(i)}}(\mathbf{z}_{k-1}),\, \mathbf{z} \right\rangle + \frac{\gamma}{2} ||\mathbf{z} - \mathbf{w}_{t-1}||^2 + \frac{\kappa}{2} ||\mathbf{z} - \mathbf{y}_{r-1}||^2. \qquad (3.12)$$
      3. All machines reach consensus by averaging local updates through another round of communication:
         $$\mathbf{z}_k \leftarrow \frac{1}{m} \sum_{i=1}^{m} \mathbf{z}_k^{(i)}. \qquad (3.13)$$
    end for
    Update $\mathbf{x}_r \leftarrow \mathbf{z}_K$.
    Compute $\alpha_r \in (0,1)$ such that $\alpha_r^2 = (1-\alpha_r)\alpha_{r-1}^2 + \gamma\alpha_k/(\gamma+\kappa)$, and compute
       $$\mathbf{y}_r = \mathbf{x}_r + \left( \frac{\alpha_{r-1}(1-\alpha_{r-1})}{\alpha_r + \alpha_{r-1}^2} \right) (\mathbf{x}_r - \mathbf{x}_{r-1}). \qquad (3.14)$$
  end for
  Update $\mathbf{w}_t \leftarrow \mathbf{x}_r$.
end for
Output: $\mathbf{w}_T$ is the approximate solution.
```

for-loop), AIDE (the intermediate **for**-loop) and DANE (the inner **for**-loop) respectively. Compared to EMSO, DANE adds a gradient correction term to (3.17) which can be compute efficiently with one round of communication. On top of that, AIDE uses the idea of universal catalyst [Lin et al., 2015] and adds an extra quadratic term to improve the strong-convexity of the objective for faster convergence, i.e., in order to solve (3.16), AIDE solves multiple instances of the "augmented large minibatch" problems of the form

$$\min_{\mathbf{w}\in\Omega} \bar{f}_{t,r}(\mathbf{w}) := \phi_{I_t}(\mathbf{w}) + \frac{\gamma}{2}||\mathbf{w}-\mathbf{w}_{t-1}||^2 + \frac{\kappa}{2}||\mathbf{w}-\mathbf{y}_{r-1}||^2 \tag{3.15}$$

with carefully chosen extrapolation points $\mathbf{y}_{r-1}$. At each DANE iteration, we perform two rounds of communication, one for averaging the local gradients, and one for averaging the local updates, and the amount of data we communicate per round has the same size of the predictor.

To sum up, in Algorithm 2, we have introduced two levels of inexactness. First, we only approximately solve the "large minibatch" subproblem (3.16) in each outer loop; results from the previous section guarantee the convergence of this approach. Second, we only approximately solve the local subproblems (3.12) to sufficient accuracy in each inner loop; the analysis of "inexact DANE" (for the non-stochastic setting) provides guarantee for this approach [Reddi et al., 2016], and enables us to use state-of-the-art SGD methods (e.g., SVRG Johnson and Zhang, 2013, Xiao and Zhang, 2014) for solving local subproblems. Overall, we obtain a convergent algorithm for distributed stochastic convex optimization.

We now present detailed analysis for the computation/communication complexity of Algorithm 2 for stochastic quadratic problems, and compare it with related methods in the literature.

3.4.1 Efficiency of MP-DANE

We present the main results of this section (full analysis is deferred to Appendix 3.7.10), which show that with careful choices of the minibatch size and the desired accuracy in each level of approximate solution, MP-DANE achieves both communication and computation efficiency with the optimal sample complexity. Interestingly, the choices of parameters differ in two regimes which are separated by an "optimal" minibatch size (also denoted as $b_{\text{mp-dane}}$ in the main text)

$$b^* = \frac{n(\varepsilon)L^2}{32m^2\beta^2B^2\log(md)}.$$

Theorem 21 (Efficiency of MP-DANE for $b \le b^*$)**.** *Set the parameters in Algorithm 2 as follows:*

$$\begin{aligned}
&\textit{(outer loop)} && b \le b^* = \frac{n(\varepsilon)L^2}{32m^2\beta^2B^2\log(md)}, \quad T = \frac{n(\varepsilon)}{bm}, \quad \gamma = \frac{\sqrt{8n(\varepsilon)}L}{bmB}, \\
&\textit{(intermediate loop)} && \kappa = 0, \quad R = 1, \\
&\textit{(inner loop)} && \theta = \frac{1}{6}, \quad K = \mathcal{O}\left(\log n(\varepsilon)\right).
\end{aligned}$$

Then we have $\mathbb{E}\left[\phi\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t\right) - \phi(\mathbf{w}_*)\right] \le \frac{\sqrt{40}BL}{\sqrt{n(\varepsilon)}} = \mathcal{O}\left(\varepsilon\right).$

Moreover, Algorithm 2 can be implemented with $\widetilde{\mathcal{O}}\left(\frac{n(\varepsilon)}{bm}\right)$ *rounds of communication, and each machine performs* $\widetilde{\mathcal{O}}\left(\frac{n(\varepsilon)}{m}\right)$ *vector operations in total, where the notation* $\widetilde{\mathcal{O}}(\cdot)$ *hides poly-logarithmic dependences on* $n(\varepsilon)$*.*

When we choose $b = b^*$*, Algorithm 1 can be implemented with* $\widetilde{\mathcal{O}}\left(\frac{m\beta^2B^2}{L^2}\right)$ *rounds of communication,* $\widetilde{\mathcal{O}}\left(\frac{n(\varepsilon)}{bm}\right)$ *vector operations, and* $\mathcal{O}\left(\frac{n(\varepsilon)L^2}{m^2\beta^2B^2}\right)$ *memory for each machine.*

We comment on the choice of parameters. For sample efficiency, we fix the sample size $n(\varepsilon)$ and number of machines m, and so we can tradeoff the local minibatch size b and the total number of outer iterations T, maintaining $bT = \frac{n(\varepsilon)}{m}$. For any b, the regularization parameters in the "large minibatch" problem is set to $\gamma = \sqrt{\frac{8T}{bm} \cdot \frac{L}{B}} = \frac{\sqrt{8n(\varepsilon)}L}{bmB}$ according to Theorem 19. When $b \le b^*$, we note that (3.35) can be satisfied with $\kappa = 0$ and there is no need for acceleration by AIDE ($R = 1$). Then the values of θ and K follow from Lemma 8.

Remark 2. *The above theorem suggests that in the regime of $b \le b^*$, we only need to have logarithmic number of DANE iterations for solving each "large minibatch" problem, and logarithmic number of passes over the local data during each DANE iteration. We present experimental results validating our theory in Appendix 3.6.*

The next theorem shows that when we use a large minibatch size b in Algorithm 2, we can still satisfy the condition (3.35) by adding extra regularization ($\kappa > 0$), and then apply accelerated DANE.

Theorem 3 (Efficiency of MP-DANE for $b \ge b^*$)**.** *Set the parameters in Algorithm 2 as follows:*

$$\begin{aligned}
&\textit{(outer loop)} \qquad b \ge b^* = \frac{n(\varepsilon)L^2}{32m^2\beta^2B^2\log(md)}, \quad T = \frac{n(\varepsilon)}{bm}, \quad \gamma = \frac{\sqrt{8n(\varepsilon)}L}{bmB}, \\
&\textit{(intermediate loop)} \qquad \kappa = 16\beta\sqrt{\frac{\log(dm)}{b}} - \gamma, \quad R = \mathcal{O}\left(\frac{b^{1/4}m^{1/2} \cdot \beta^{1/2}B^{1/2}}{n(\varepsilon)^{1/4} \cdot L^{1/2}} \log n(\varepsilon)\right), \\
&\textit{(inner loop)} \qquad \theta = \frac{1}{6}, \quad K = \mathcal{O}\left(\log n(\varepsilon)\right).
\end{aligned}$$

Then we have $\mathbb{E}\left[\phi\left(\frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t\right) - \phi(\mathbf{w}_*)\right] \le \frac{\sqrt{40}BL}{\sqrt{n(\varepsilon)}} = \mathcal{O}(\varepsilon)$.

Moreover, Algorithm 2 can be implemented with $\widetilde{\mathcal{O}}\left(\frac{n(\varepsilon)^{3/4} \cdot \beta^{1/2}B^{1/2}}{b^{3/4}m^{1/2} \cdot L^{1/2}}\right)$ rounds of communication, and each machine performs $\widetilde{\mathcal{O}}\left(\frac{b^{1/4}n(\varepsilon)^{3/4} \cdot \beta^{1/2}B^{1/2}}{m^{1/2} \cdot L^{1/2}}\right)$ vector operations in total, where the notation $\widetilde{\mathcal{O}}(\cdot)$ hides poly-logarithmic dependences on $n(\varepsilon)$.

	Samples	Communication	Computation	Memory
$1 \le b \le b^*$	$n(\varepsilon)$	$n(\varepsilon)/mb$	$n(\varepsilon)/m$	b
$b = b^*$	$n(\varepsilon)$	$B^2 m$	$n(\varepsilon)/m$	$n(\varepsilon)/(m^2 B^2)$
$b^* < b \le b_{\max}$	$n(\varepsilon)$	$B^{1/2} n(\varepsilon)^{3/4}/(m^{1/2} b^{3/4})$	$B^{1/2} n(\varepsilon)^{3/4} b^{1/4}/m^{1/2}$	b

Table 3.2: Summary of resources required by MP-DANE for distributed stochastic convex optimization, in units of vector operations/communications/memory per machine, ignoring constants and log-factors. Here $b^* \asymp n(\varepsilon)/(m^2 B^2)$, and $b_{\max} = n(\varepsilon)/m$.

3.4.2 Two regimes of multiple resource tradeoffs

From the above analysis, we summarized in Table 3.2 the resources required by MP-DANE. We observe two interesting regimes, separated by the minibatch size $b^* \asymp n(\varepsilon)/(m^2 B^2)$, that present different tradeoffs between communication, computation and memory.

- When $1 \le b \le b^*$, the computation complexity remains $\widetilde{\mathcal{O}}\left(n(\varepsilon)/m\right)$ which is independent of b. This means we always achieve *near-linear speedup* in this regime. Moreover, there is a tradeoff between communication and memory: the communication complexity decreases, while the memory cost increases as the minibatch size b increases, both at the linear rate. Thus in this regime, we can trade communication for memory without affecting computation.
- When $b^* < b \le b_{\max}$, the computation starts to increase with b at the rate $b^{1/4}$ which is slower than linear, while the communication cost continues to decrease at the rate $b^{3/4}$ which is also slower than linear. Thus in this regime, we can trade communication for computation and memory.

3.5 Communication-efficient distributed minibatch-prox with SVRG

We now apply the theoretical results of minibatch-prox to the distributed stochastic learning setting, and propose a novel algorithm that is both communication and computation

Algorithm 3 Minibatch-prox with DSVRG for distributed stochastic convex optimization.

```
Initialize $\mathbf{w}_0 = \mathbf{0}$.
for $t = 1, 2, \dots, T$ do
    % Outer loop performs minibatch-prox.
    Each machine $i$ draws a minibatch $I_t^{(i)}$ of $b$ samples from the underlying data distribution, and split $I_t^{(i)}$ to $p_i$ batches of size $b/p_i$: $B_1^{(i)}, B_2^{(i)}, ..., B_{p_i}^{(i)}$
    Initialize $\mathbf{z}_0 \leftarrow \mathbf{w}_{t-1}$,   $\mathbf{x}_0 \leftarrow \mathbf{w}_{t-1}$,   $j \leftarrow 1$,   $s \leftarrow 1$
    for $k = 1, 2, \dots, K$ do
        1. All machines perform one round of communication to compute the average gradient:
           $$\nabla\phi_{I_t}(\mathbf{z}_{k-1}) \leftarrow \frac{1}{m}\sum_{i=1}^{m} \nabla\phi_{I_t^{(i)}}(\mathbf{z}_{k-1})$$
        2. Machine $j$ performs stochastic updates by going through $B_s^{(j)}$ once without replacement:
           $$\mathbf{x}_r \leftarrow \mathbf{x}_{r-1} - \eta\left(\nabla\ell(\mathbf{x}_{r-1}, \xi_l) - \nabla\ell(\mathbf{z}_{k-1}, \xi_l) + \nabla\phi_{I_t}(\mathbf{z}_{k-1}) + \gamma(\mathbf{x}_{r-1} - \mathbf{w}_{t-1})\right)$$
           for $\xi_l \in B_s^{(j)}$.
        3. Machine $j$ update $\mathbf{z}_k$:
           $$\mathbf{z}_k \leftarrow \frac{1}{|B_s^{(j)}|}\sum_{r=0}^{|B_s^{(j)}|} \mathbf{x}_r,$$
           and broadcast $\mathbf{z}_k$ to other machines.
        4. Update indices: $s \leftarrow s + 1$,
        if $s > p_j$ then
            $s \leftarrow 1$,   $j \leftarrow j + 1$.
        end if
    end for
    Update $\mathbf{w}_t \leftarrow \mathbf{z}_K$.
end for
Output: $\mathbf{w}_T$ is the approximate solution.
```

efficient, and being able to explore trade-offs between memory and communication efficiency.

Suppose we have m machines in a distributed environment. For each outer loop of our algorithm, each machine i draws a minibatch $I_t^{(i)}$ of b samples independently from other machines, and denote $I_t = \cup_{i=1}^{m} I_t^{(i)}$ which contains bm samples. To apply the minibatch-prox algorithm from the previous section, we need to find an approximate solution to the following problem:

$$\min_{\mathbf{w}} \ \widetilde{f}_t(\mathbf{w}) := \phi_{I_t}(\mathbf{w}) + \frac{\gamma}{2}||\mathbf{w} - \mathbf{w}_{t-1}||^2. \tag{3.16}$$

Since the objective (3.16) involves functions from different machines, we use distributed optimization algorithms for solving it. In Li et al., the authors proposed a simple algorithm EMSO to approximately solve (3.16), where each machine first solve its own local objective, i.e.,

$$\mathbf{w}_t^{(i)} = \operatorname*{argmin}_{\mathbf{w}} \ \phi_{I_t^{(i)}} + \frac{\gamma}{2}||\mathbf{w} - \mathbf{w}_{t-1}||^2, \tag{3.17}$$

and then all machines average their local solutions via one round of communication: $\mathbf{w}_t = \frac{1}{m}\sum_{i=1}^m \mathbf{w}_t^{(i)}$.

We note that this can be considered as the "one-shot-averaging" approach [Zhang et al., 2012] for solving (3.16). Although this approach was shown to work well empirically, no convergence guarantee for the original stochastic objective (3.1) was provided by Li et al.. Here we instead use the distributed SVRG (DSVRG) algorithm [Lee et al., 2017a, Shamir, 2016] to approximately solve (3.16), as DSVRG enjoys excellent communication and computation cost when the problem is well conditioned (cf. Table 7.1).[3]

We detail our algorithm, named MP-DSVRG (minibatch-prox with DSVRG), in Algorithm 3. The algorithm consists of two nested loops, where t, k are iteration counters for minibatch-prox (the outer **for**-loop), and DSVRG (the inner **for**-loop) respectively. In each outer loop, each machine draws a minibatch $I_t^{(i)}$ to form the objective (3.16), which will be solved approximately by the inner loops. Moreover, each machine splits its local dataset into p_i batches: $I^{(i)} = \cup_{j=1}^{p_i} B_j^{(i)}$. In each inner loop, all machines communicate to calculate the

3. It is also possible to equip minibatch-prox with other communication-efficient distributed optimization algorithms, for example in Appendix 3.4, we present a minibatch-prox DANE (MP-DANE) algorithm which uses the accelerated DANE method for solving (3.16).

global gradient (averaged local gradients) of (3.16), and then one of the machines j picks a local batch $B_s^{(j)}$ to perform the stochastic updates, where the local batch contains enough samples such that one pass of stochastic updates on $B_s^{(j)}$ decrease the objective quickly. We perform two rounds of communication in each inner loop, one for computing the global gradient, and one for broadcasting the new predictor obtained by a machine j. As we will show in the next section, by carefully choosing the parameters, we will obtain a convergent algorithm for distributed stochastic convex optimization with better efficiency guarantees than previous methods.

We now present detailed analysis for the computation/communication complexity of Algorithm 3 for stochastic quadratic problems, and compare it with related methods in the literature. Throughout this section, we have $\ell(\mathbf{w}, \xi) = \frac{1}{2}(\mathbf{w}^\top \mathbf{x} - y)^2$ where $\xi = (\mathbf{x}, y)$. We assume that $\ell(\mathbf{w}, \xi)$ is β-smooth and L-Lipschitz in $\mathbf{w}$,[4] and we would like to learn a predictor that is competitive to all predictors with norm at most B. Note that each $\ell(\mathbf{w}, \xi)$ is only weakly convex.

3.5.1 Efficiency of MP-DSVRG

For the distributed stochastic convex optimization problems, we are concerned with efficiency in terms of sample, communication, computation and memory. Recall that for convex L-Lipshitz, B-bounded problems, to learn a predictor $\widehat{\mathbf{w}}$ with ε-generalization error, i.e., $\mathbb{E}\left[\phi(\widehat{\mathbf{w}}) - \phi(\mathbf{w}_*)\right] \leq \varepsilon$, we require the sample size to be at least $n(\varepsilon) = \mathcal{O}(L^2B^2/\varepsilon^2)$. This sample complexity matches the worst case lower bound, and can be achieved by vanilla SGD.

The theorem below shows that with careful choices of parameters in the outer and inner loops, MP-DSVRG achieves both communication and computation efficiency with the optimal sample complexity.

4. We can equivalently assume $||\mathbf{x}||^2 \leq \beta$ and y is bounded.

Theorem 22 (Efficiency of MP-DSVRG)**.** *Set the parameters in Algorithm 3 as follows:*

$$\textit{(outer loop)} \qquad T = \frac{n(\varepsilon)}{bm}, \quad \gamma = \frac{\sqrt{8n(\varepsilon)}L}{bmB}, \quad p_i = \mathcal{O}\left(\frac{\sqrt{n(\varepsilon)}L}{\beta mB}\right)$$

$$\textit{(inner loop)} \qquad K = \mathcal{O}\left(\log n(\varepsilon)\right).$$

Then we have $\mathbb{E}\left[\phi\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t\right) - \phi(\mathbf{w}_*)\right] \le \frac{\sqrt{40}BL}{\sqrt{n(\varepsilon)}} = \mathcal{O}\left(\varepsilon\right).$

Moreover, Algorithm 3 can be implemented with $\mathcal{O}\left(\frac{n(\varepsilon)}{bm}\log n(\varepsilon)\right)$ *rounds of communication, and each machine performs* $\mathcal{O}\left(\frac{n(\varepsilon)}{m}\log n(\varepsilon)\right)$ *vector operations in total.*

We comment on the choice of parameters. For sample efficiency, we fix the sample size $n(\varepsilon)$ and number of machines m, and so we can tradeoff the local minibatch size b and the total number of outer iterations T, maintaining $bT = \frac{n(\varepsilon)}{m}$. For any b, the regularization parameters in the "large minibatch" problem is set to $\gamma = \sqrt{\frac{8T}{bm}} \cdot \frac{L}{B} = \frac{\sqrt{8n(\varepsilon)}L}{bmB}$ according to Theorem 19. Moreover, we choose the number of batches p_i in each local machine in a way that performing one pass of stochastic updates over a single batch by without-replacement sampling is sufficient to reduce the objective by a constant factor.

3.6 Experiments

Table 3.3: List of datasets used in the experiments.

Name	#Samples	#Features	loss
codrna	271,617	8	logistic
covtype	581,012	54	logistic
kddcup99	1,131,571	127	logistic
year	463,715	90	squared

In this section we present empirical results to support our theoretical analysis of MP-DANE. We perform least squares regression and classification on several publicly available

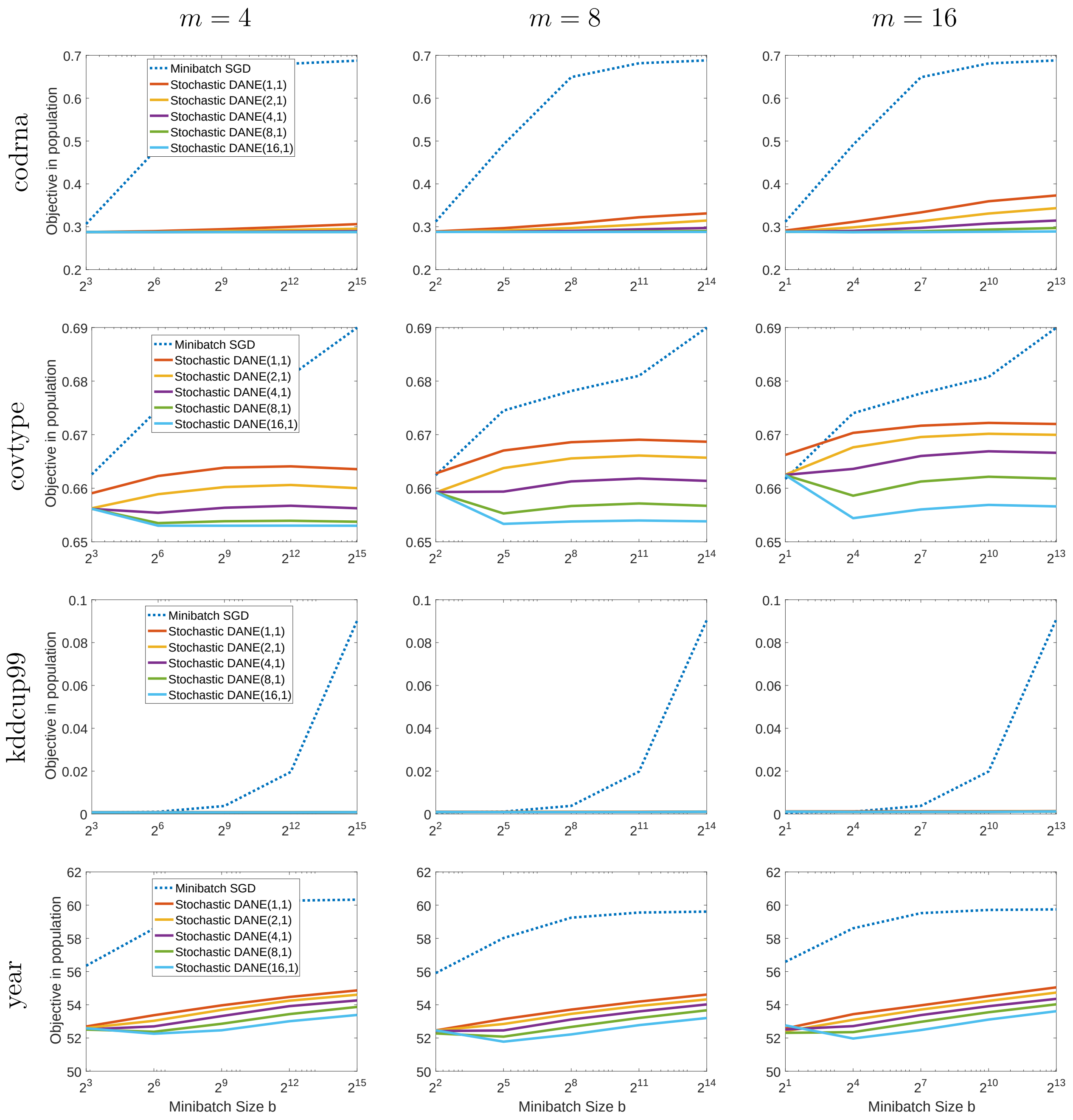


Figure 3.2: Illustration of the convergence properties of MP-DANE, for different minibatch size b, number of machines m, and number of DANE iterations K.

datasets[5]; the statistics of these datasets and the corresponding losses are summarized in Table 5.2. For each dataset, we randomly select half of the samples for training, and the remaining samples are used for estimating the stochastic objective.

For MP-DANE, we use SAGA [Defazio et al., 2014] to solve each local DANE subproblem (3.12) and fix the number of SAGA steps to b (i.e., we just make one pass over the local data), while varying the number of DANE rounds K over $\{1, 2, 4, 8, 16\}$. For simplicity, we do not use catalyst acceleration and set $R = 1$ and $\kappa = 0$ in all experiments. Our experiments simulate a distributed environment with m machines, for $m = 4, 8, 16$. We conduct a simple comparison with minibatch SGD. Stepsizes for SAGA and minibatch SGD are set based on the smoothness parameter of the loss.

We plot in Figure 3.2 the estimated population objective vs. minibatch size b for different parameters. We make the following observations.

- For minibatch SGD, as b increases, the objective often increases quickly, this is because minibatch SGD can not uses large minibatch sizes while preserving sample efficiency.
- For MP-DANE, the objective increases much more slowly as b increases. This demonstrates the effectiveness of minibatch-prox for using large minibatch sizes.
- Running more iterations of DANE often helps, but with diminishing returns. This validates our theory that only a near-constant number of DANE iterations is needed for solving the large minibatch objective, without affecting the sample efficiency.

5. `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

3.7 Proofs of technical results

3.7.1 Proof of Lemma 13

Proof. Observe that (3.4) implies $\gamma_t(\mathbf{w}_{t-1} - \mathbf{w}_t)$ is a subgradient at $\mathbf{w}_t$ of the sum of $\phi_{I_t}(\mathbf{w})$ and the indicator function of Ω (which has value 0 in Ω and ∞ otherwise), and thus we have for any $\mathbf{w} \in \Omega$ that

$$\phi_{I_t}(\mathbf{w}) - \phi_{I_t}(\mathbf{w}_t) \geq \gamma_t \left\langle \mathbf{w}_{t-1} - \mathbf{w}_t,\, \mathbf{w} - \mathbf{w}_t \right\rangle + \frac{\lambda}{2}||\mathbf{w} - \mathbf{w}_t||^2. \tag{3.18}$$

For any $\mathbf{w} \in \Omega$, we can bound its distance to $\mathbf{w}_{t-1}$ as

$$\begin{aligned}
||\mathbf{w}_{t-1} - \mathbf{w}||^2 &= ||\mathbf{w}_{t-1} - \mathbf{w}_t + \mathbf{w}_t - \mathbf{w}||^2 \\
&= ||\mathbf{w}_{t-1} - \mathbf{w}_t||^2 + 2\left\langle \mathbf{w}_{t-1} - \mathbf{w}_t,\, \mathbf{w}_t - \mathbf{w} \right\rangle + ||\mathbf{w}_t - \mathbf{w}||^2 \\
&\geq ||\mathbf{w}_{t-1} - \mathbf{w}_t||^2 + \frac{2}{\gamma_t}\left(\phi_{I_t}(\mathbf{w}_t) - \phi_{I_t}(\mathbf{w})\right) + \frac{\lambda}{\gamma_t}||\mathbf{w} - \mathbf{w}_t||^2 + ||\mathbf{w}_t - \mathbf{w}||^2 \\
&= \frac{\lambda + \gamma_t}{\gamma_t}||\mathbf{w}_t - \mathbf{w}||^2 + \frac{2}{\gamma_t}\left(\phi_{I_t}(\mathbf{w}_t) - \phi_{I_t}(\mathbf{w})\right) + ||\mathbf{w}_{t-1} - \mathbf{w}_t||^2
\end{aligned}$$

where we have used (3.18) in the first inequality. Rearranging the terms yields the desired result. □

3.7.2 Proof of Lemma 14

The following lemma, which is essentially shown by Shalev-Shwartz et al. [2009, Theorem 6], characterizes the convergence of the empirical loss to the population counterpart for the (approximate) regularized empirical risk minimizer.

Lemma 4. *Let the instantaneous function $\ell(\mathbf{w}, \xi)$ be L-Lipschitz and λ-strongly convex in*

$\mathbf{w}$. *Consider the following regularized ERM problem with sample set* $Z = \{\xi_1, \ldots, \xi_n\}$*:*

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}\in\Omega}{\operatorname{argmin}}\ \widehat{F}(\mathbf{w}) \qquad \textit{where} \quad \widehat{F}(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}, \xi_i) + r(\mathbf{w}),$$

and the regularizer $r(\mathbf{w})$ *is* γ*-strongly convex. Denote by* $G(\mathbf{w}) = \mathbb{E}_\xi\left[\ell(\mathbf{w}, \xi)\right]$ *and* $\widehat{G}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}, \xi_i)$ *the expected and the empirical losses respectively.*

1. *For the regularized empirical risk minimizer* $\widehat{\mathbf{w}}$*, we have*

$$\left|\mathbb{E}_Z\left[G(\widehat{\mathbf{w}}) - \widehat{G}(\widehat{\mathbf{w}})\right]\right| \le \frac{4L^2}{(\lambda+\gamma)n}.$$

2. *If for any given dataset* Z*, a possibly randomized algorithm* $\mathcal{A}$ *minimizes* $\widehat{F}(\mathbf{w})$ *up to an error of* η*, i.e.,* $\mathcal{A}$ *returns an approximate solution* $\widetilde{\mathbf{w}}$ *such that* $\mathbb{E}_{\mathcal{A}}\left[\widehat{F}(\widetilde{\mathbf{w}}) - \widehat{F}(\widehat{\mathbf{w}})\right] \le \eta$*, we have*

$$\left|\mathbb{E}_{Z,\mathcal{A}}\left[G(\widetilde{\mathbf{w}}) - \widehat{G}(\widehat{\mathbf{w}})\right]\right| \le \frac{4L^2}{(\lambda+\gamma)n} + \sqrt{\frac{2L^2\eta}{\lambda+\gamma}}.$$

Proof. We prove the lemma by a stability argument.

Exact ERM Denote by $Z^{(i)}$ the sample set that is identical to Z except that the i-th sample ξ_i is replaced by another random sample ξ_i', by $\widehat{F}^{(i)}(\mathbf{w})$ the empirical objective defined using $Z^{(i)}$, i.e.,

$$\widehat{F}^{(i)}(\mathbf{w}) := \frac{1}{n}\left(\sum_{j\neq i} \ell(\mathbf{w}, \xi_i) + \ell(\mathbf{w}, \xi_i')\right) + r(\mathbf{w}),$$

and by $\widehat{\mathbf{w}}^{(i)} = \operatorname{argmin}_{\mathbf{w}\in\Omega}\ \widehat{F}^{(i)}(\mathbf{w})$ the empirical risk minimizer of $\widehat{F}^{(i)}(\mathbf{w})$.

By the definition of the empirical objectives, we have

$$\begin{aligned}
\widehat{F}(\widehat{\mathbf{w}}^{(i)}) - \widehat{F}(\widehat{\mathbf{w}}) &= \frac{\ell(\widehat{\mathbf{w}}^{(i)}, \xi_i) - \ell(\widehat{\mathbf{w}}, \xi_i)}{n} + \frac{\sum_{j\neq i} \ell(\widehat{\mathbf{w}}^{(i)}, \xi_i) - \ell(\widehat{\mathbf{w}}, \xi_i)}{n} + r(\widehat{\mathbf{w}}^{(i)}) - r(\widehat{\mathbf{w}}) \\
&= \frac{\ell(\widehat{\mathbf{w}}^{(i)}, \xi_i) - \ell(\widehat{\mathbf{w}}, \xi_i)}{n} + \frac{\ell(\widehat{\mathbf{w}}, \xi_i') - \ell(\widehat{\mathbf{w}}^{(i)}, \xi_i')}{n} + \left(\widehat{F}^{(i)}(\widehat{\mathbf{w}}^{(i)}) - \widehat{F}^{(i)}(\widehat{\mathbf{w}})\right) \\
&\leq \frac{\left|\ell(\widehat{\mathbf{w}}^{(i)}, \xi_i) - \ell(\widehat{\mathbf{w}}, \xi_i)\right|}{n} + \frac{\left|\ell(\widehat{\mathbf{w}}, \xi_i') - \ell(\widehat{\mathbf{w}}^{(i)}, \xi_i')\right|}{n} \\
&\leq \frac{2L}{n} ||\widehat{\mathbf{w}}^{(i)} - \widehat{\mathbf{w}}|| \qquad (3.19)
\end{aligned}$$

where we have used the fact that $\widehat{\mathbf{w}}^{(i)}$ is the minimizer of $\widehat{F}^{(i)}(\mathbf{w})$ in the first inequality, and the L-Lipschitz continuity of $\ell(\mathbf{w}, \xi)$ in the second inequality.

On the other hand, it follows from the $(\lambda + \gamma)$-strong convexity of $\widehat{F}(\mathbf{w})$ that

$$\widehat{F}(\widehat{\mathbf{w}}^{(i)}) - \widehat{F}(\widehat{\mathbf{w}}) \geq \frac{(\lambda + \gamma)}{2} ||\widehat{\mathbf{w}}^{(i)} - \widehat{\mathbf{w}}||^2. \qquad (3.20)$$

Combining (3.19) and (3.20) yields $||\widehat{\mathbf{w}}^{(i)} - \widehat{\mathbf{w}}|| \leq \frac{4L}{(\lambda+\gamma)n}$.

Again, by the L-Lipschitz continuity of $\ell(\mathbf{w}, \xi)$, we have that for any sample ξ that

$$\left|\ell(\widehat{\mathbf{w}}, \xi) - \ell(\widehat{\mathbf{w}}^{(i)}, \xi)\right| \leq L ||\widehat{\mathbf{w}}^{(i)} - \widehat{\mathbf{w}}|| \leq \frac{4L^2}{(\lambda + \gamma)n}. \qquad (3.21)$$

Since Z and $Z^{(i)}$ are both i.i.d. sample sets, we have

$$\mathbb{E}_Z\left[G(\widehat{\mathbf{w}})\right] = \mathbb{E}_{Z^{(i)}}\left[G(\widehat{\mathbf{w}}^{(i)})\right] = \mathbb{E}_{Z^{(i)}\cup\{\xi_i\}}\left[\ell(\widehat{\mathbf{w}}^{(i)}, \xi_i)\right].$$

As this holds for all $i = 1, \ldots, n$, we can also write

$$\mathbb{E}_Z\left[G(\widehat{\mathbf{w}})\right] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{Z^{(i)}\cup\{\xi_i\}}\left[\ell(\widehat{\mathbf{w}}^{(i)}, \xi_i)\right]. \qquad (3.22)$$

On the other hand, we have

$$\mathbb{E}_Z\left[\widehat{G}(\widehat{\mathbf{w}})\right] = \mathbb{E}_Z\left[\frac{1}{n}\sum_{i=1}^n \ell(\widehat{\mathbf{w}}, \xi_i)\right] = \frac{1}{n}\sum_{i=1}^n \mathbb{E}_Z\left[\ell(\widehat{\mathbf{w}}, \xi_i)\right]. \tag{3.23}$$

Combining (3.22) and (3.23) and using the stability (3.21), we obtain

$$\mathbb{E}_Z\left[G(\widehat{\mathbf{w}}) - \widehat{G}(\widehat{\mathbf{w}})\right] = \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{Z\cup\{\xi_i'\}}\left[\ell(\widehat{\mathbf{w}}^{(i)}, \xi_i) - \ell(\widehat{\mathbf{w}}, \xi_i)\right] \in \left[-\frac{4L^2}{(\lambda+\gamma)n}, \frac{4L^2}{(\lambda+\gamma)n}\right].$$

Inexact ERM For the approximate solution $\widetilde{\mathbf{w}}$, due to the $(\lambda+\gamma)$-strong convexity of $\widehat{F}(\mathbf{w})$, we have

$$\mathbb{E}_{\mathcal{A}}||\widetilde{\mathbf{w}} - \widehat{\mathbf{w}}||^2 \le \frac{2}{\lambda+\gamma}\mathbb{E}_{\mathcal{A}}\left[\widehat{F}(\widetilde{\mathbf{w}}) - \widehat{F}(\widehat{\mathbf{w}})\right] \le \frac{2\eta}{\lambda+\gamma},$$

and thus $\mathbb{E}_{\mathcal{A}}||\widetilde{\mathbf{w}} - \widehat{\mathbf{w}}|| \le \sqrt{\frac{2\eta}{\lambda+\gamma}}$ by the fact that $\mathbb{E}x^2 \ge (\mathbb{E}x)^2$ for any random variable x.

It then follows from the Lipschitz continuity of $G(\mathbf{w})$ that

$$\mathbb{E}_{\mathcal{A}}\left|G(\widetilde{\mathbf{w}}) - G(\widehat{\mathbf{w}})\right| \le L \cdot \mathbb{E}_{\mathcal{A}}||\widetilde{\mathbf{w}} - \widehat{\mathbf{w}}|| \le \sqrt{\frac{2L^2\eta}{\lambda+\gamma}}.$$

Finally, we have by the triangle inequality and the stability of exact ERM that

$$\begin{aligned}\left|\mathbb{E}_{Z,\mathcal{A}}\left[G(\widetilde{\mathbf{w}}) - \widehat{G}(\widehat{\mathbf{w}})\right]\right| &\le \mathbb{E}_Z\left[\mathbb{E}_{\mathcal{A}}\left|G(\widetilde{\mathbf{w}}) - G(\widehat{\mathbf{w}})\right|\right] + \left|\mathbb{E}_Z\left[G(\widehat{\mathbf{w}}) - \widehat{G}(\widehat{\mathbf{w}})\right]\right| \\ &\le \sqrt{\frac{2L^2\eta}{\lambda+\gamma}} + \frac{4L^2}{(\lambda+\gamma)n}.\end{aligned}$$

□

Then Lemma 14 follows from the fact that that our stochastic objective (3.8) is equipped with L-Lipschitz, λ-strongly convex loss $\phi(\mathbf{w})$ and γ_t-strongly convex regularizer $\frac{\gamma_t}{2}||\mathbf{w} - \mathbf{w}_{t-1}||^2$.

3.7.3 Proof of Lemma 15

Proof. We have by Lemma 14 that

$$\left|\mathbb{E}_{I_t}\left[\phi_{I_t}(\mathbf{w}_t) - \phi(\mathbf{w}_t)\right]\right| \leq \frac{4L^2}{(\lambda+\gamma_t)b}.$$

Take expectation of (3.6) over the random sampling of I_t and we obtain

$$\begin{aligned}
\frac{\lambda+\gamma_t}{\gamma_t}\mathbb{E}_{I_t}||\mathbf{w}_t - \mathbf{w}||^2 &\leq ||\mathbf{w}_{t-1} - \mathbf{w}||^2 - \frac{2}{\gamma_t}\left(\mathbb{E}_{I_t}\left[\phi_{I_t}(\mathbf{w}_t)\right] - \phi(\mathbf{w})\right) \\
&= ||\mathbf{w}_{t-1} - \mathbf{w}||^2 - \frac{2}{\gamma_t}\left(\mathbb{E}_{I_t}\left[\phi_{I_t}(\mathbf{w}_t) - \phi(\mathbf{w}_t)\right] + \mathbb{E}_{I_t}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w})\right]\right) \\
&\leq ||\mathbf{w}_{t-1} - \mathbf{w}||^2 - \frac{2}{\gamma_t}\mathbb{E}_{I_t}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w})\right] + \frac{2}{\gamma_t}\left|\mathbb{E}_{I_t}\left[\phi_{I_t}(\mathbf{w}_t) - \phi(\mathbf{w}_t)\right]\right| \\
&\leq ||\mathbf{w}_{t-1} - \mathbf{w}||^2 - \frac{2}{\gamma_t}\mathbb{E}_{I_t}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w})\right] + \frac{8L^2}{\gamma_t(\lambda+\gamma_t)b}.
\end{aligned}$$

□

3.7.4 Proof of Theorem 16

Proof. When $\ell(\mathbf{w}, \xi)$ is weakly convex (i.e., $\lambda = 0$), we further set $\gamma_t = \gamma$ for all $t \geq 1$. Applying Lemma 15 with $\mathbf{w} = \mathbf{w}_*$ yields

$$\mathbb{E}_{I_t}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w}_*)\right] \leq \frac{\gamma}{2}\left(||\mathbf{w}_{t-1} - \mathbf{w}_*||^2 - \mathbb{E}_{I_t}||\mathbf{w}_t - \mathbf{w}_*||^2\right) + \frac{4L^2}{\gamma b}. \tag{3.24}$$

Summing (3.24) for $t = 1, \ldots, T$ yields

$$\sum_{t=1}^{T}\mathbb{E}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w}_*)\right] \leq \frac{\gamma}{2}||\mathbf{w}_0 - \mathbf{w}_*||^2 + \frac{4L^2T}{\gamma b}.$$

Minimizing the RHS over γ gives the optimal choice

$$\gamma = \sqrt{\frac{8T}{b}} \cdot \frac{L}{||\mathbf{w}_0 - \mathbf{w}_*||},$$

with a corresponding regret

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w}_*)\right] \leq \frac{\sqrt{8}L}{\sqrt{bT}}||\mathbf{w}_0 - \mathbf{w}_*||.$$

As a result, by returning the uniform average $\widehat{\mathbf{w}}_T = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$, we have due to the convexity of $\phi(\mathbf{w})$ that

$$\mathbb{E}\left[\phi(\widehat{\mathbf{w}}_T) - \phi(\mathbf{w}_*)\right] \leq \frac{\sqrt{8}L}{\sqrt{bT}}||\mathbf{w}_0 - \mathbf{w}_*||.$$

□

3.7.5 Proof of Theorem 17

Proof. Let $\ell(\mathbf{w}, \xi)$ be λ-strongly convex for some $\lambda > 0$. Applying Lemma 15 with $\mathbf{w} = \mathbf{w}_*$ yields

$$\mathbb{E}_{I_t}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w}_*)\right] \leq \left(\frac{\gamma_t}{2}||\mathbf{w}_{t-1} - \mathbf{w}_*||^2 - \frac{\lambda + \gamma_t}{2}\mathbb{E}_{I_t}||\mathbf{w}_t - \mathbf{w}_*||^2\right) + \frac{4L^2}{(\lambda + \gamma_t)b}. \quad (3.25)$$

Setting $\gamma_t = \frac{\lambda(t-1)}{2}$ for $t = 1, \ldots,$[6], the above inequality becomes

$$\begin{aligned}\mathbb{E}_{I_t}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w}_*)\right] &\leq \left(\frac{\lambda(t-1)}{4}||\mathbf{w}_{t-1} - \mathbf{w}_*||^2 - \frac{\lambda(t+1)}{4}\mathbb{E}_{I_t}||\mathbf{w}_t - \mathbf{w}_*||^2\right) + \frac{8L^2}{\lambda b(t+1)} \\ &\leq \left(\frac{\lambda(t-1)}{4}||\mathbf{w}_{t-1} - \mathbf{w}_*||^2 - \frac{\lambda(t+1)}{4}\mathbb{E}_{I_t}||\mathbf{w}_t - \mathbf{w}_*||^2\right) + \frac{8L^2}{\lambda bt},\end{aligned}$$

6. This choice is inspired by the stepsize rule of Lacoste-Julien et al. [2012] for stochastic gradient descent.

and therefore

$$t \cdot \mathbb{E}_{I_t}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w}_*)\right] \leq \frac{\lambda}{4}\left((t-1)t||\mathbf{w}_{t-1} - \mathbf{w}_*||^2 - t(t+1)\mathbb{E}_{I_t}||\mathbf{w}_t - \mathbf{w}_*||^2\right) + \frac{8L^2}{\lambda b}.$$

Summing this inequality for $t = 1, \ldots, T$ yields

$$\sum_{t=1}^{T} t \cdot \mathbb{E}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w}_*)\right] \leq \frac{8L^2 T}{\lambda b}.$$

As a result, by returning the weighted average $\widehat{\mathbf{w}}_T = \frac{2}{T(T+1)}\sum_{t=1}^{T} t\mathbf{w}_t$, we have due to the convexity of $\phi(\mathbf{w})$ that $\phi(\widehat{\mathbf{w}}_T) \leq \frac{2}{T(T+1)}\sum_{t=1}^{T} t \cdot \phi(\mathbf{w}_t)$ and

$$\mathbb{E}\left[\phi(\widehat{\mathbf{w}}_T) - \phi(\mathbf{w}_*)\right] \leq \frac{2}{T(T+1)}\sum_{t=1}^{T} t \cdot \mathbb{E}\left[\phi(\mathbf{w}_t) - \phi(\mathbf{w}_*)\right] \leq \frac{16L^2}{\lambda b(T+1)}.$$

□

3.7.6 Proof of Lemma 18

Proof. Due to the $(\lambda + \gamma_t)$-strong convexity of $\widetilde{f}_t(\mathbf{w})$, we have

$$\mathbb{E}_{\mathcal{A}}||\widetilde{\mathbf{w}}_t - \bar{\mathbf{w}}_t||^2 \leq \frac{2}{\lambda + \gamma_t}\mathbb{E}_{\mathcal{A}}\left[\widetilde{f}_t(\widetilde{\mathbf{w}}_t) - \widetilde{f}_t(\bar{\mathbf{w}}_t)\right] \leq \frac{2\eta_t}{\lambda + \gamma_t}.$$

Applying Lemma 13 to the exact minimizer $\bar{\mathbf{w}}_t$ yields

$$\phi_{I_t}(\bar{\mathbf{w}}_t) - \phi_{I_t}(\mathbf{w}) \leq \frac{\gamma_t}{2}||\widetilde{\mathbf{w}}_{t-1} - \mathbf{w}||^2 - \frac{\lambda + \gamma_t}{2}||\bar{\mathbf{w}}_t - \mathbf{w}||^2.$$

Therefore, for the t-th iteration, we have

$$
\begin{aligned}
&\mathbb{E}_{I_t,\mathcal{A}}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w})\right] \\
&= \mathbb{E}_{I_t,\mathcal{A}}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi_{I_t}(\bar{\mathbf{w}}_t)\right] + \mathbb{E}_{I_t}\left[\phi_{I_t}(\bar{\mathbf{w}}_t) - \phi_{I_t}(\mathbf{w})\right] \\
&\leq \frac{4L^2}{(\lambda+\gamma_t)b} + \sqrt{\frac{2L^2\eta_t}{\lambda+\gamma_t}} + \frac{\gamma_t}{2}||\widetilde{\mathbf{w}}_{t-1} - \mathbf{w}||^2 - \frac{\lambda+\gamma_t}{2}\mathbb{E}_{I_t}||\bar{\mathbf{w}}_t - \mathbf{w}||^2 \\
&\leq \frac{4L^2}{(\lambda+\gamma_t)b} + \sqrt{\frac{2L^2\eta_t}{\lambda+\gamma_t}} + \frac{\gamma_t}{2}||\widetilde{\mathbf{w}}_{t-1} - \mathbf{w}||^2 - \frac{\lambda+\gamma_t}{2}\mathbb{E}_{I_t,\mathcal{A}}\left(||\widetilde{\mathbf{w}}_t - \mathbf{w}|| - ||\widetilde{\mathbf{w}}_t - \bar{\mathbf{w}}_t||\right)^2 \\
&\leq \frac{4L^2}{(\lambda+\gamma_t)b} + \sqrt{\frac{2L^2\eta_t}{\lambda+\gamma_t}} + \frac{\gamma_t}{2}||\widetilde{\mathbf{w}}_{t-1} - \mathbf{w}||^2 - \frac{\lambda+\gamma_t}{2}\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t - \mathbf{w}||^2 \\
&\qquad\qquad + (\lambda+\gamma_t)\cdot\mathbb{E}_{I_t,\mathcal{A}}\left[||\widetilde{\mathbf{w}}_t - \bar{\mathbf{w}}_t||\cdot||\widetilde{\mathbf{w}}_t - \mathbf{w}||\right] \\
&\leq \frac{4L^2}{(\lambda+\gamma_t)b} + \sqrt{\frac{2L^2\eta_t}{\lambda+\gamma_t}} + \frac{\gamma_t}{2}||\widetilde{\mathbf{w}}_{t-1} - \mathbf{w}||^2 - \frac{\lambda+\gamma_t}{2}\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t - \mathbf{w}||^2 \\
&\qquad\qquad + (\lambda+\gamma_t)\sqrt{\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t - \bar{\mathbf{w}}_t||^2}\cdot\sqrt{\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t - \mathbf{w}||^2} \\
&\leq \frac{4L^2}{(\lambda+\gamma_t)b} + \sqrt{\frac{2L^2\eta_t}{\lambda+\gamma_t}} + \frac{\gamma_t}{2}||\widetilde{\mathbf{w}}_{t-1} - \mathbf{w}||^2 - \frac{\lambda+\gamma_t}{2}\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t - \mathbf{w}||^2 \\
&\qquad\qquad + \sqrt{2(\lambda+\gamma_t)\eta_t}\cdot\sqrt{\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t - \mathbf{w}||^2}
\end{aligned}
$$

where we have applied Lemma 4 to the approximate minimizer $\widetilde{\mathbf{w}}_t$ in the first inequality, used the triangle inequality $||\bar{\mathbf{w}}_t - \mathbf{w}|| \geq |||\widetilde{\mathbf{w}}_t - \mathbf{w}|| - ||\widetilde{\mathbf{w}}_t - \bar{\mathbf{w}}_t|||$ in the second inequality, dropped a negative term in the third inequality, and used the Cauchy-Schwarz inequality for random variables in the fourth inequality. □

3.7.7 Proof of Theorem 19

When $\ell(\mathbf{w}, \xi)$ is weakly convex (i.e., $\lambda = 0$), set $\gamma_t = \gamma$ for all $t \geq 1$ as in exact minibatch-prox. Then summing (3.11) for $t = 1, \dots, T$ yields

$$\sum_{t=1}^{T} \mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*)\right] + \frac{\gamma}{2}\mathbb{E}||\widetilde{\mathbf{w}}_T - \mathbf{w}_*||^2 \leq \frac{\gamma}{2}||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||^2 + \frac{4L^2T}{\gamma b} + \sum_{t=1}^{T}\sqrt{\frac{2L^2\eta_t}{\gamma}} + \sum_{t=1}^{T}\sqrt{2\gamma\eta_t} \cdot \sqrt{\mathbb{E}||\widetilde{\mathbf{w}}_t - \mathbf{w}_*||^2} \qquad (3.26)$$

where the expectation is taken over random sampling and the randomness of $\mathcal{A}$ in the first T iterations. To resolve the recursion, we need the following lemma by Schmidt et al. [2011].

Lemma 5. *Assume that the non-negative sequence $\{u_T\}$ satisfies the following recursion for all $T \geq 1$:*

$$u_T^2 \leq S_T + \sum_{t=1}^{T} \lambda_t u_t,$$

with S_T an increasing sequence, $S_0 \geq u_0^2$ and $\lambda_t \geq 0$ for all t. Then, for all $T \geq 1$, we have

$$u_T \leq \frac{1}{2}\sum_{t=1}^{T}\lambda_t + \left(S_T + \left(\frac{1}{2}\sum_{t=1}^{T}\lambda_t\right)^2\right)^{\frac{1}{2}} \leq \sqrt{S_T} + \sum_{t=1}^{T}\lambda_t.$$

We are now ready to prove Theorem 19.

Proof. **Bounding $\sqrt{\mathbb{E}||\widetilde{\mathbf{w}}_t - \mathbf{w}_*||^2}$.** Dropping the $\sum_{t=1}^{T}\mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*)\right]$ term from (3.26) which is non-negative due to the optimality of $\mathbf{w}_*$, we obtain

$$\mathbb{E}||\widetilde{\mathbf{w}}_T - \mathbf{w}_*||^2 \leq ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||^2 + \frac{8L^2T}{\gamma^2 b} + \sum_{t=1}^{T}\sqrt{\frac{8L^2\eta_t}{\gamma^3}} + \sum_{t=1}^{T}\sqrt{\frac{8\eta_t}{\gamma}} \cdot \sqrt{\mathbb{E}||\widetilde{\mathbf{w}}_t - \mathbf{w}_*||^2}.$$

Now apply Lemma 5 (using $u_T = \sqrt{\mathbb{E}||\widetilde{\mathbf{w}}_T - \mathbf{w}_*||^2}$, $S_T = ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||^2 + \frac{8L^2T}{\gamma^2 b} + \sum_{t=1}^T \sqrt{\frac{8L^2\eta_t}{\gamma^3}}$, and $\lambda_t = \sqrt{\frac{8\eta_t}{\gamma}}$) and the fact that $\sqrt{x+y} \le \sqrt{x} + \sqrt{y}$ for $x,\, y \ge 0$, we have

$$\sqrt{\mathbb{E}||\widetilde{\mathbf{w}}_T - \mathbf{w}_*||^2} \le ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*|| + \sqrt{\frac{8L^2T}{\gamma^2 b}} + \sum_{t=1}^T \sqrt{\frac{8\eta_t}{\gamma}} + \sqrt{\sum_{t=1}^T \sqrt{\frac{8L^2\eta_t}{\gamma^3}}}$$

We have thus bounded the sequence of $\sqrt{\mathbb{E}||\widetilde{\mathbf{w}}_T - \mathbf{w}_*||^2}$ by a non-negative increasing sequence.

Bounding function values. Dropping the $\mathbb{E}||\widetilde{\mathbf{w}}_T - \mathbf{w}_*||^2$ term from (3.26) which is non-negative, we obtain

$$\begin{aligned}
&\sum_{t=1}^T \mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*)\right] \\
\le\ & \frac{\gamma}{2}||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||^2 + \frac{4L^2T}{\gamma b} + \sum_{t=1}^T \sqrt{\frac{2L^2\eta_t}{\gamma}} + \sum_{t=1}^T \sqrt{2\eta_t\gamma} \cdot \sqrt{\mathbb{E}||\widetilde{\mathbf{w}}_t - \mathbf{w}_*||^2} \\
\le\ & \frac{\gamma}{2}||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||^2 + \frac{4L^2T}{\gamma b} + \sum_{t=1}^T \sqrt{\frac{2L^2\eta_t}{\gamma}} + \left(\sum_{t=1}^T \sqrt{2\eta_t\gamma}\right) \cdot \max_{1\le t\le T} \sqrt{\mathbb{E}||\widetilde{\mathbf{w}}_t - \mathbf{w}_*||^2} \\
\le\ & \frac{\gamma}{2}||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||^2 + \frac{4L^2T}{\gamma b} + \sum_{t=1}^T \sqrt{\frac{2L^2\eta_t}{\gamma}} \\
& + \left(\sum_{t=1}^T \sqrt{2\eta_t\gamma}\right) \cdot \left(||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*|| + \sqrt{\frac{8L^2T}{\gamma^2 b}} + \sum_{t=1}^T \sqrt{\frac{8\eta_t}{\gamma}} + \sqrt{\sum_{t=1}^T \sqrt{\frac{8L^2\eta_t}{\gamma^3}}}\right). \qquad (3.27)
\end{aligned}$$

To achieve the same order of regret as in exact minibatch-prox, we require that η_t decays with t, and in particular

$$\eta_t \le \min\left(c_1\left(\frac{T}{b}\right)^{\frac{1}{2}},\, c_2\left(\frac{T}{b}\right)^{\frac{3}{2}}\right) \cdot \frac{L||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||}{t^{2+2\delta}} \qquad (3.28)$$

for some $\delta > 0$. Note that η_t has the unit of function value. Let $c := \sum_{i=1}^\infty \frac{1}{i^{1+\delta}} \le \frac{1+\delta}{\delta}$

which only depends on δ (as a concrete example, we have $c = \frac{\pi^2}{6}$ when $\delta = 2$).

Using the choice of $\gamma = \sqrt{\frac{8T}{b}} \cdot \frac{L}{||\mathbf{w}_0 - \mathbf{w}_*||}$, we obtain from (3.28) that

$$\begin{aligned}
\sum_{t=1}^{T} \sqrt{\frac{8\eta_t}{\gamma}} &= \sum_{t=1}^{T} \sqrt{\sqrt{\frac{8b}{T}} \cdot \frac{||\mathbf{w}_0 - \mathbf{w}_*||}{L} \cdot \eta_t} \le 8^{\frac{1}{4}} c_1^{\frac{1}{2}} ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*|| \sum_{t=1}^{T} \frac{1}{t^{1+\delta}} \\
&\le 8^{\frac{1}{4}} c_1^{\frac{1}{2}} c ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||, \\
\sum_{t=1}^{T} \sqrt{\frac{8L^2\eta_t}{\gamma^3}} &= \sum_{t=1}^{T} \sqrt{\sqrt{\frac{b^3}{8T^3}} \cdot \frac{||\mathbf{w}_0 - \mathbf{w}_*||^3}{L} \cdot \eta_t} \le 8^{-\frac{1}{4}} c_2^{\frac{1}{2}} ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||^2 \sum_{t=1}^{T} \frac{1}{t^{1+\delta}} \\
&\le 8^{-\frac{1}{4}} c_2^{\frac{1}{2}} c ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||^2.
\end{aligned}$$

Continuing from (3.27) and substituting in the value of γ, we have

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*)\right] &\le \sqrt{\frac{8T}{b}} \cdot L||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*|| + \frac{\gamma}{2} \sum_{t=1}^{T} \sqrt{\frac{8L^2\eta_t}{\gamma^3}} \\
&\quad + \frac{\gamma}{2} \left(\sum_{t=1}^{T} \sqrt{\frac{8\eta_t}{\gamma}} \right) \cdot \left(2||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*|| + \sum_{t=1}^{T} \sqrt{\frac{8\eta_t}{\gamma}} + \sqrt{\sum_{t=1}^{T} \sqrt{\frac{8L^2\eta_t}{\gamma^3}}} \right) \\
&= \sqrt{\frac{8T}{b}} \cdot L||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*|| + \sqrt{\frac{2T}{b}} \cdot \frac{L}{||\mathbf{w}_0 - \mathbf{w}_*||} \cdot 8^{-\frac{1}{4}} c_2^{\frac{1}{2}} c ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||^2 \\
&\quad + \sqrt{\frac{2T}{b}} \cdot \frac{L}{||\mathbf{w}_0 - \mathbf{w}_*||} \cdot 8^{\frac{1}{4}} c_1^{\frac{1}{2}} c ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*|| \times \\
&\quad \left(2||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*|| + 8^{\frac{1}{4}} c_1^{\frac{1}{2}} c ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*|| + \sqrt{8^{-\frac{1}{4}} c_2^{\frac{1}{2}} c ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||^2} \right) \\
&= c_3 \sqrt{\frac{T}{b}} \cdot L||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||.
\end{aligned}$$

The suboptimality of $\widehat{\mathbf{w}}_T$ is then due to the convexity of $\phi(\mathbf{w})$:

$$\mathbb{E}\left[\phi(\widehat{\mathbf{w}}_T) - \phi(\mathbf{w}_*)\right] \le \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*)\right] = \frac{c_3 L ||\widetilde{\mathbf{w}}_0 - \mathbf{w}_*||}{\sqrt{bT}}.$$

$\square$

3.7.8 Proof of Theorem 20

Proof. We have by Lemma 18 that

$$\mathbb{E}_{I_t,\mathcal{A}}\left[\phi(\widetilde{\mathbf{w}}_t)-\phi(\mathbf{w}_*)\right] \le \frac{\lambda(t-1)}{4}||\widetilde{\mathbf{w}}_{t-1}-\mathbf{w}_*||^2 - \frac{\lambda(t+1)}{4}\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t-\mathbf{w}_*||^2 + \frac{8L^2}{\lambda b(t+1)} + \sqrt{\frac{4L^2\eta_t}{\lambda(t+1)}} + \sqrt{\lambda(t+1)\eta_t}\cdot\sqrt{\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t-\mathbf{w}_*||^2}.$$

Relaxing the $\frac{1}{t+1}$ to $\frac{1}{t}$ on the RHS, and multiplying both sides by t, we further obtain

$$t\cdot\mathbb{E}_{I_t,\mathcal{A}}\left[\phi(\widetilde{\mathbf{w}}_t)-\phi(\mathbf{w}_*)\right] \le \frac{\lambda(t-1)t}{4}||\widetilde{\mathbf{w}}_{t-1}-\mathbf{w}_*||^2 - \frac{\lambda t(t+1)}{4}\mathbb{E}_{I_t,\mathcal{A}}||\widetilde{\mathbf{w}}_t-\mathbf{w}_*||^2 + \frac{8L^2}{\lambda b} + \sqrt{\frac{4L^2 t\eta_t}{\lambda}} + \sqrt{\lambda t\eta_t}\cdot\sqrt{\mathbb{E}_{I_t,\mathcal{A}}\left[t(t+1)||\widetilde{\mathbf{w}}_t-\mathbf{w}_*||^2\right]}.$$

Summing this inequality for $t=1,\ldots,T$ yields

$$\sum_{t=1}^{T} t\cdot\mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t)-\phi(\mathbf{w}_*)\right] + \frac{\lambda T(T+1)}{4}\mathbb{E}||\widetilde{\mathbf{w}}_T-\mathbf{w}_*||^2 \le \frac{8L^2T}{\lambda b} + \sum_{t=1}^{T}\sqrt{\frac{4L^2t\eta_t}{\lambda}} + \sum_{t=1}^{T}\sqrt{\lambda t\eta_t}\cdot\sqrt{\mathbb{E}\left[t(t+1)||\widetilde{\mathbf{w}}_t-\mathbf{w}_*||^2\right]}. \tag{3.29}$$

Bounding $\sqrt{\mathbb{E}||\widetilde{\mathbf{w}}_t-\mathbf{w}_*||^2}$. Dropping the $\sum_{t=1}^{T} t\cdot\mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t)-\phi(\mathbf{w}_*)\right]$ term from (3.29) which is non-negative due to the optimality of $\mathbf{w}_*$, we obtain

$$\mathbb{E}\left[T(T+1)||\widetilde{\mathbf{w}}_T-\mathbf{w}_*||^2\right] \le \frac{32L^2T}{\lambda^2 b} + \sum_{t=1}^{T}\sqrt{\frac{64L^2t\eta_t}{\lambda^3}} + \sum_{t=1}^{T}\sqrt{\frac{16t\eta_t}{\lambda}}\cdot\sqrt{\mathbb{E}\left[t(t+1)||\widetilde{\mathbf{w}}_t-\mathbf{w}_*||^2\right]}.$$

Applying Lemma 5 (using $u_T=\sqrt{\mathbb{E}\left[T(T+1)||\widetilde{\mathbf{w}}_T-\mathbf{w}_*||^2\right]}$, $S_T=\frac{32L^2T}{\lambda^2 b}+\sum_{t=1}^{T}\sqrt{\frac{64L^2t\eta_t}{\lambda^3}}$, and $\lambda_t = \sqrt{\frac{16 t \eta_t}{\lambda}}$), we have

$$\sqrt{\mathbb{E}\left[T(T+1)||\widetilde{\mathbf{w}}_T - \mathbf{w}_*||^2\right]} \le \sqrt{\frac{32L^2T}{\lambda^2 b}} + \sum_{t=1}^{T} \sqrt{\frac{16t\eta_t}{\lambda}} + \sqrt{\sum_{t=1}^{T} \sqrt{\frac{64L^2 t\eta_t}{\lambda^3}}}.$$

Bounding function values. Dropping the $\mathbb{E}||\widetilde{\mathbf{w}}_T - \mathbf{w}_*||^2$ term from (3.29) which is non-negative, we obtain

$$\begin{aligned}\sum_{t=1}^{T} t \cdot \mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*)\right] &\le \frac{8L^2T}{\lambda b} + \sum_{t=1}^{T} \sqrt{\frac{4L^2 t\eta_t}{\lambda}} \\ &+ \left(\sum_{t=1}^{T} \sqrt{\lambda t \eta_t}\right) \cdot \left(\sqrt{\frac{32L^2T}{\lambda^2 b}} + \sum_{t=1}^{T} \sqrt{\frac{16t\eta_t}{\lambda}} + \sqrt{\sum_{t=1}^{T} \sqrt{\frac{64L^2 t\eta_t}{\lambda^3}}}\right). \quad (3.30)\end{aligned}$$

To achieve the same order of regret as in exact minibatch-prox, we require that η_t decays with t, and in particular

$$\eta_t \le \min\left(c_1\left(\frac{T}{b}\right), c_2\left(\frac{T}{b}\right)^2\right) \cdot \frac{L^2}{t^{3+2\delta}\lambda} \quad (3.31)$$

for some $\delta > 0$. Note that η_t has the unit of function value. Let $c := \sum_{i=1}^{\infty} \frac{1}{i^{1+\delta}} \le \frac{1+\delta}{\delta}$. Then (3.31) ensures that

$$\sum_{t=1}^{T} \sqrt{\frac{t\eta_t}{\lambda}} \le c\sqrt{c_1}\sqrt{\frac{L^2T}{\lambda^2 b}}, \qquad \text{and} \qquad \sum_{t=1}^{T} \sqrt{\frac{L^2 t\eta_t}{\lambda^3}} \le c\sqrt{c_2} \cdot \frac{L^2T}{\lambda^2 b}.$$

Continuing from (3.30), we have

$$\begin{aligned}\sum_{t=1}^{T} t \cdot \mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*)\right] &\leq \frac{8L^2T}{\lambda b} + 2c\sqrt{c_2} \cdot \frac{L^2T}{\lambda b} \\ &\quad + c\sqrt{c_1}\sqrt{\frac{L^2T}{b}} \left(\sqrt{\frac{32L^2T}{\lambda^2 b}} + 4c\sqrt{c_1}\sqrt{\frac{L^2T}{\lambda^2 b}} + \sqrt[4]{64c^2c_2}\sqrt{\frac{L^2T}{\lambda^2 b}} \right) \\ &= \frac{c_3}{2} \cdot \frac{L^2T}{\lambda b}.\end{aligned}$$

In view of the convexity of $\phi(\mathbf{w})$, by returning the weighted average $\widehat{\mathbf{w}}_T = \frac{2}{T(T+1)} \sum_{t=1}^{T} t\widetilde{\mathbf{w}}_t$, we have

$$\mathbb{E}\left[\phi(\widehat{\mathbf{w}}_T) - \phi(\mathbf{w}_*)\right] \leq \frac{2}{T(T+1)} \sum_{t=1}^{T} t \cdot \mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*)\right] \leq \frac{c_3 L^2}{\lambda b (T+1)}.$$

$\square$

3.7.9 Connection to minibatch stochastic gradient descent

To see the connection between minibatch-prox and minibatch SGD, note that if we solve the linearized minibatch problem exactly, we obtain the minibatch stochastic gradient descent algorithm:

$$\widetilde{\mathbf{w}}_t = \operatorname*{argmin}_{\mathbf{w} \in \Omega} \; \phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}) + \nabla \left\langle \phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}),\, \mathbf{w} - \widetilde{\mathbf{w}}_{t-1} \right\rangle + \frac{\gamma_t}{2} ||\mathbf{w} - \widetilde{\mathbf{w}}_{t-1}||^2.$$

Following Cotter et al. [2011], we assume that $\ell(\mathbf{w}, \xi)$ is β-smooth:

$$||\nabla \ell(\mathbf{w}, \xi) - \nabla \ell(\mathbf{w}', \xi)|| \leq \beta ||\mathbf{w} - \mathbf{w}'||, \qquad \forall \mathbf{w}, \mathbf{w}' \in \Omega.$$

We then have the following guarantee for each iterate of minbatch SGD.

Proposition 6. *For iteration t of minibatch SGD, we have*

$$\mathbb{E}_{I_t}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*)\right] \le \frac{2L^2}{(\gamma_t - \beta)b} + \frac{\gamma_t - \lambda}{2}||\mathbf{w}_* - \widetilde{\mathbf{w}}_{t-1}||^2 - \frac{\gamma_t}{2}\mathbb{E}_{I_t}||\mathbf{w}_* - \widetilde{\mathbf{w}}_t||^2. \qquad (3.32)$$

Proof. Our proof closely follows that of Cotter et al. [2011].

Due to the smoothness of ϕ, we have that

$$\begin{aligned}
\phi(\widetilde{\mathbf{w}}_t) &\le \phi(\widetilde{\mathbf{w}}_{t-1}) + \langle \nabla\phi(\widetilde{\mathbf{w}}_{t-1}), \widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}\rangle + \frac{\beta}{2}||\widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}||^2 \\
&\le \phi(\widetilde{\mathbf{w}}_{t-1}) + \left\langle \nabla\phi(\widetilde{\mathbf{w}}_{t-1}) - \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}), \widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}\right\rangle + \frac{\beta}{2}||\widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}||^2 \\
&\qquad + \left\langle \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}), \widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}\right\rangle \\
&= \phi(\widetilde{\mathbf{w}}_{t-1}) + ||\nabla\phi(\widetilde{\mathbf{w}}_{t-1}) - \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1})|| \cdot ||\widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}|| + \frac{\beta}{2}||\widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}||^2 \\
&\qquad + \left\langle \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}), \widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}\right\rangle \\
&\le \phi(\widetilde{\mathbf{w}}_{t-1}) + \frac{1}{2(\gamma_t - \beta)}||\nabla\phi(\widetilde{\mathbf{w}}_{t-1}) - \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1})||^2 + \frac{\gamma_t - \beta}{2}||\widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}||^2 \\
&\qquad + \frac{\beta}{2}||\widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}||^2 + \left\langle \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}), \widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}\right\rangle \\
&= \phi(\widetilde{\mathbf{w}}_{t-1}) + \frac{1}{2(\gamma_t - \beta)}||\nabla\phi(\widetilde{\mathbf{w}}_{t-1}) - \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1})||^2 + \frac{\gamma_t}{2}||\widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}||^2 \\
&\qquad + \left\langle \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}), \widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}\right\rangle \qquad (3.33)
\end{aligned}$$

where we have used the Cauchy-Schwarz inequality in the second inequality, and the inequality $xy \le \frac{x^2}{2\alpha} + \frac{\alpha y^2}{2}$ in the third inequality.

Now, since $\widetilde{\mathbf{w}}_t$ is the minimizer of the γ_t-strongly convex function

$$\frac{\gamma_t}{2}||\mathbf{w} - \widetilde{\mathbf{w}}_{t-1}||^2 + \left\langle \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}), \mathbf{w} - \widetilde{\mathbf{w}}_{t-1}\right\rangle$$

in Ω, we have according to Lemma 13 (replacing the local objective with its linear approximation) that

$$\begin{aligned}
&\frac{\gamma_t}{2}||\mathbf{w}_* - \widetilde{\mathbf{w}}_{t-1}||^2 + \left\langle \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}),\, \mathbf{w}_* - \widetilde{\mathbf{w}}_{t-1} \right\rangle \\
\geq\ &\frac{\gamma_t}{2}||\widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1}||^2 + \left\langle \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}),\, \widetilde{\mathbf{w}}_t - \widetilde{\mathbf{w}}_{t-1} \right\rangle + \frac{\gamma_t}{2}||\mathbf{w}_* - \widetilde{\mathbf{w}}_t||^2.
\end{aligned}$$

Substituting this into (3.33) gives

$$\begin{aligned}
\phi(\widetilde{\mathbf{w}}_t) \leq\ &\phi(\widetilde{\mathbf{w}}_{t-1}) + \frac{1}{2(\gamma_t - \beta)}||\nabla\phi(\widetilde{\mathbf{w}}_{t-1}) - \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1})||^2 + \frac{\gamma_t}{2}||\mathbf{w}_* - \widetilde{\mathbf{w}}_{t-1}||^2 \\
&+ \left\langle \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1}),\, \mathbf{w}_* - \widetilde{\mathbf{w}}_{t-1} \right\rangle - \frac{\gamma_t}{2}||\mathbf{w}_* - \widetilde{\mathbf{w}}_t||^2.
\end{aligned}$$

Taking expectation of this inequality over the random sampling of I_t further leads to

$$\begin{aligned}
\mathbb{E}_{I_t}\left[\phi(\widetilde{\mathbf{w}}_t)\right] \leq\ &\phi(\widetilde{\mathbf{w}}_{t-1}) + \frac{1}{2(\gamma_t - \beta)}\mathbb{E}_{I_t}||\nabla\phi(\widetilde{\mathbf{w}}_{t-1}) - \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1})||^2 + \frac{\gamma_t}{2}||\mathbf{w}_* - \widetilde{\mathbf{w}}_{t-1}||^2 \\
&+ \langle \nabla\phi(\widetilde{\mathbf{w}}_{t-1}),\, \mathbf{w}_* - \widetilde{\mathbf{w}}_{t-1} \rangle - \frac{\gamma_t}{2}\mathbb{E}_{I_t}||\mathbf{w}_* - \widetilde{\mathbf{w}}_t||^2 \\
\leq\ &\phi(\mathbf{w}_*) + \frac{1}{2(\gamma_t - \beta)}\mathbb{E}_{I_t}||\nabla\phi(\widetilde{\mathbf{w}}_{t-1}) - \nabla\phi_{I_t}(\widetilde{\mathbf{w}}_{t-1})||^2 \\
&+ \frac{\gamma_t - \lambda}{2}||\mathbf{w}_* - \widetilde{\mathbf{w}}_{t-1}||^2 - \frac{\gamma_t}{2}\mathbb{E}_{I_t}||\mathbf{w}_* - \widetilde{\mathbf{w}}_t||^2 \quad (3.34)
\end{aligned}$$

where in the second inequality we have used the fact that

$$\phi(\mathbf{w}_*) \geq \phi(\widetilde{\mathbf{w}}_{t-1}) + \langle \nabla\phi(\widetilde{\mathbf{w}}_{t-1}),\, \mathbf{w}_* - \widetilde{\mathbf{w}}_{t-1} \rangle + \frac{\lambda}{2}||\mathbf{w}_* - \widetilde{\mathbf{w}}_{t-1}||^2$$

due to the convexity of $\phi(\mathbf{w})$.

On the other hand, let $I_t = \{\xi_1, \dots, \xi_b\}$, we have

$$\begin{aligned}
& \mathbb{E}_{I_t} ||\nabla\phi(\mathbf{w}) - \nabla\phi_{I_t}(\mathbf{w})||^2 \\
= {} & \mathbb{E}_{I_t} ||\nabla\phi(\mathbf{w}) - \frac{1}{b}\sum_{i=1}^{b} \nabla\ell(\mathbf{w}, \xi_i)||^2 \\
= {} & \mathbb{E}_{I_t} ||\frac{1}{b}\sum_{i=1}^{b} \left(\nabla\phi(\mathbf{w}) - \nabla\ell(\mathbf{w}, \xi_i)\right)||^2 \\
= {} & \frac{1}{b^2}\sum_{i=1}^{b} \mathbb{E}_{\xi_i} ||\nabla\phi(\mathbf{w}) - \nabla\ell(\mathbf{w}, \xi_i)||^2 + \frac{1}{b^2}\sum_{i\neq j} \mathbb{E}_{I_t} \left\langle \nabla\phi(\mathbf{w}) - \nabla\ell(\mathbf{w}, \xi_i),\, \nabla\phi(\mathbf{w}) - \nabla\ell(\mathbf{w}, \xi_j)\right\rangle \\
= {} & \frac{1}{b} \cdot \mathbb{E}_{\xi} ||\nabla\phi(\mathbf{w}) - \nabla\ell(\mathbf{w}, \xi)||^2 \\
\leq {} & \frac{4L^2}{b}
\end{aligned}$$

where we used the fact that the samples are i.i.d. in the fourth equality, and the fact that $||\nabla\phi(\mathbf{w})||,\ ||\nabla\ell(\mathbf{w}, \xi)|| \leq L$ in the last inequality. Continuing from (3.34) yields the desired result. □

Comparing this result to (3.24) and (3.25), we observe that minibatch SGD has a similar recursion to that exact minibatch-prox, except the appearance of β in the denominator of the "stability" term. We now show that this difference leads to significant difference in convergence rate.

Let $\ell(\mathbf{w}, \xi)$ be weakly convex ($\lambda = 0$), and $\gamma_t = \gamma$ for all $t \geq 1$. Summing (3.32) over $t = 1, \dots, T$ gives

$$\sum_{t=1}^{T} \mathbb{E}\left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*)\right] \leq \frac{2L^2 T}{(\gamma - \beta)b} + \frac{\gamma}{2}||\mathbf{w}_* - \widetilde{\mathbf{w}}_0||^2.$$

Minimizing the RHS over γ gives

$$\gamma = \beta + \sqrt{\frac{4T}{b}} \cdot \frac{L}{||\mathbf{w}_* - \widetilde{\mathbf{w}}_0||},$$

which leads to

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E} \left[\phi(\widetilde{\mathbf{w}}_t) - \phi(\mathbf{w}_*) \right] \le \frac{2L||\mathbf{w}_* - \widetilde{\mathbf{w}}_0||}{\sqrt{bT}} + \frac{\beta||\mathbf{w}_* - \widetilde{\mathbf{w}}_0||^2}{2T}.$$

So we obtain the familiar $\mathcal{O}\left(\frac{1}{\sqrt{bT}} + \frac{1}{T}\right)$ rate for minibatch SGD.

3.7.10 Theoretical analysis of MP-DANE

In order to fully analyze Algorithm 2, we need several auxiliary lemmas that characterize the iteration complexity of solving the local problem (3.12) by prox-SVRG [Xiao and Zhang, 2014], the large minibatch problem (3.16) by DANE [Shamir et al., 2014] and AIDE [Reddi et al., 2016].

Some auxiliary lemmas

First, we apply prox-SVRG to the local problem (3.12), pushing all terms but $\phi_{I_t^{(i)}}(\mathbf{z})$ in to the proximal operator. The benefit of this approach (as opposed to using plain SVRG Johnson and Zhang, 2013) is that the smoothness parameter that determines the iteration complexity is simply β, same results hold when applying prox-SAGA [Defazio et al., 2014] as well. For sampling without replacement SVRG, the current analysis works only for plain SVRG, so we quote the results from [Shamir, 2016].

Lemma 7 (Iteration complexity of SVRG for (3.12))**.** *For any target accuracy $\theta > 0$, with* *initialization* $\mathbf{z}_{k-1}$*, prox-SVRG outputs* $\mathbf{z}_k^{(i)}$ *such that* $||\mathbf{z}_k^{(i)} - \mathbf{z}_k^{(i)*}|| \le \theta||\mathbf{z}_{k-1} - \mathbf{z}_k^{(i)*}||$ *after*

$$\mathcal{O}\left(\left(b + \frac{\beta}{\gamma+\kappa}\right) \cdot \log \frac{(\beta+\gamma+\kappa)}{(\gamma+\kappa)\theta^2}\right)$$

vector operations, and sampling without replacement SVRG outputs $\mathbf{z}_k^{(i)}$ *such that* $||\mathbf{z}_k^{(i)} - \mathbf{z}_k^{(i)*}|| \le \theta||\mathbf{z}_{k-1} - \mathbf{z}_k^{(i)*}||$ *after*

$$\mathcal{O}\left(\left(b + \frac{\beta+\kappa}{\gamma+\kappa}\right) \cdot \log \frac{(\beta+\gamma+\kappa)}{(\gamma+\kappa)\theta^2}\right)$$

vector operations.

Proof. Observe that the objective (3.12) by $f_k^{(i)}(\mathbf{z})$, which is an quadratic function of $\mathbf{z}$ with the Hessian matrix $H_i = \nabla^2\phi_{I_t^{(i)}}(\mathbf{z}) + (\gamma+\kappa)\mathbf{I} \succeq (\gamma+\kappa)\mathbf{I}$. As a result, the suboptimality of $\mathbf{z}_k^{(i)}$ is

$$\epsilon_{\text{final}} = f_k^{(i)}(\mathbf{z}_k^{(i)}) - f_k^{(i)}(\mathbf{z}_k^{(i)*}) = \frac{1}{2}\left(\mathbf{z}_k^{(i)} - \mathbf{z}_k^{(i)*}\right)^\top H_i \left(\mathbf{z}_k^{(i)} - \mathbf{z}_k^{(i)*}\right) \ge \frac{\gamma+\kappa}{2}||\mathbf{z}_k^{(i)} - \mathbf{z}_k^{(i)*}||^2.$$

To satisfy the requirement of $||\mathbf{z}_k^{(i)} - \mathbf{z}_k^{(i)*}|| \le \theta||\mathbf{z}_{k-1} - \mathbf{z}_k^{(i)*}||$, we require

$$\epsilon_{\text{final}} \le \frac{(\gamma+\kappa)\theta^2}{2}||\mathbf{z}_{k-1} - \mathbf{z}_k^{(i)*}||^2.$$

On the other hand, when initializing from z_{k-1}, the initial suboptimality is

$$\epsilon_{\text{init}} = f_k^{(i)}(\mathbf{z}_{k-1}) - f_k^{(i)}(\mathbf{z}_k^{(i)*}) \le \frac{\sigma_{\max}(H_i)}{2}||\mathbf{z}_{k-1} - \mathbf{z}_k^{(i)*}||^2 \le \frac{\beta+\gamma+\kappa}{2}||\mathbf{z}_{k-1} - \mathbf{z}_k^{(i)*}||^2.$$

Therefore, it suffices to have

$$\frac{\epsilon_{\text{init}}}{\epsilon_{\text{final}}} = \frac{(\beta+\gamma+\kappa)}{(\gamma+\kappa)\theta^2}.$$

Noting that $\phi_{I_t^{(i)}}(\mathbf{z})$ is the sum of b components, and each component is β-smooth while the overall function $f_k^{(i)}$ is $(\gamma + \kappa)$-strongly convex, the lemma follows directly from the convergence guarantee of prox-SVRG [Xiao and Zhang, 2014, Corollary 1], and sampling without replacement SVRG [Shamir, 2016, Theorem 4]. □

Next, we state the convergence rates of "inexact DANE" and AIDE, which can be easily derived from Reddi et al. [2016]. At the outer loop t and intermediate loop r, let $\mathbf{x}_r^* = \operatorname{argmin}_{\mathbf{w}} \bar{f}_{t,r}(\mathbf{w})$ be the exact minimizer of the "augmented large minibatch" problem (3.15), which is approximately solved by the inner DANE iterations.

Lemma 8 (Iteration Complexity of inexact DANE)**.** *Let $\theta = \frac{1}{6}$, and assume that*

$$b(\gamma + \kappa)^2 \geq 256\beta^2 \log(dm/\delta). \tag{3.35}$$

By initializing from $\mathbf{y}_{r-1}$, and setting the number of inner iterations in Algorithm 2 to be

$$K = \lceil \frac{1}{2} \log_{4/3} \frac{(\beta + \gamma + \kappa)}{(\gamma + \kappa)\eta} \rceil,$$

we have with probability $1 - \delta$ over the sample set I_t that

$$\bar{f}_{t,r}(\mathbf{x}_r) - \bar{f}_{t,r}(\mathbf{x}_r^*) \leq \eta \left(\bar{f}_{t,r}(\mathbf{y}_{r-1}) - \bar{f}_{t,r}(\mathbf{x}_r^*) \right).$$

Proof. Denote by $H_i = \nabla^2 \phi_{I_t^{(i)}}(\mathbf{z}) + (\gamma + \kappa)\mathbf{I}$ the Hessian matrix of the local objective (3.12) for machine i. Let $H = \frac{1}{m} \sum_{i=1}^m H_i$ be the Hessian matrix of the global objective (3.15), and $\widetilde{H}^{-1} = \frac{1}{m} \sum_{i=1}^m H_i^{-1}$. As our objective is quadratic, $H_i, H, \widetilde{H}^{-1}$ remain unchanged during the inner iterations. By Reddi et al. [2016, Theorem 1], we have

$$||\mathbf{z}_k - \mathbf{x}_r^*|| \leq \left(||\widetilde{H}^{-1} H - \mathbf{I}|| + \frac{\theta}{m} \sum_{i=1}^m ||H_i^{-1} H|| \right) ||\mathbf{z}_{k-1} - \mathbf{x}_r^*||. \tag{3.36}$$

Since $\nabla^2 \ell(\mathbf{w}, \xi) \leq \beta$, by Shamir et al. [2014, Lemma 2], we have with probability at least $1-\delta$ over the sample set I_t that

$$||H_i - H|| \leq \sqrt{\frac{32\beta^2 \log(dm/\delta)}{b}} =: \rho, \qquad\qquad i = 1, \ldots, m.$$

On the other hand, we have $H_i \succeq (\gamma+\kappa)\mathbf{I}$ and

$$\frac{4\rho^2}{(\gamma+\kappa)^2} = \frac{128\beta^2 \log(dm/\delta)}{b(\gamma+\kappa)^2} \leq \frac{1}{2}$$

by our assumption (3.35). By Shamir et al. [2014, Lemma 1], we have

$$||\widetilde{H}^{-1}H - \mathbf{I}|| \leq \frac{1}{2}. \tag{3.37}$$

Moreover, we have

$$\begin{aligned}
\frac{\theta}{m}\sum_{i=1}^{m}||H_i^{-1}H|| &\leq \frac{\theta}{m}\sum_{i=1}^{m}(1 + ||H_i^{-1}H - \mathbf{I}||) \\
&\leq \frac{\theta}{m}\sum_{i=1}^{m}(1 + ||H_i^{-1}||||H - H_i^{-1}||) \\
&\leq \frac{\theta}{m}\sum_{i=1}^{m}\left(1 + \frac{\rho}{\gamma+\kappa}\right) \\
&\leq \frac{\theta}{m}\sum_{i=1}^{m}\left(1 + \frac{1}{2\sqrt{2}}\right) \\
&\leq \frac{3\theta}{2} \leq \frac{1}{4}.
\end{aligned} \tag{3.38}$$

Plugging (3.37) and (3.38) into (3.36) yields

$$||\mathbf{z}_k - \mathbf{x}_r^*|| \leq \frac{3}{4}||\mathbf{z}_{k-1} - \mathbf{x}_r^*||,$$

and thus $||\mathbf{z}_K - \mathbf{x}_r^*|| \le (3/4)^K ||\mathbf{y}_{r-1} - \mathbf{x}_r^*||$. To guarantee the suboptimality in the objective $\bar{f}_{t,r}(\mathbf{w})$, we note that

$$\begin{aligned}\bar{f}_{t,r}(\mathbf{z}_K) - \bar{f}_{t,r}(\mathbf{x}_r^*) &= \frac{1}{2}(\mathbf{z}_K - \mathbf{x}_r^*)^\top H (\mathbf{z}_K - \mathbf{x}_r^*) \le \frac{\beta+\gamma+\kappa}{2}||\mathbf{z}_K - \mathbf{x}_r^*||^2 \\ &\le \left(\frac{3}{4}\right)^{2K} \frac{\beta+\gamma+\kappa}{2}||\mathbf{y}_{r-1} - \mathbf{x}_r^*||^2 \\ &\le \left(\frac{3}{4}\right)^{2K} \frac{\beta+\gamma+\kappa}{\gamma+\kappa}\left(\bar{f}_{t,r}(\mathbf{y}_{r-1}) - \bar{f}_{t,r}(\mathbf{x}_r^*)\right)\end{aligned}$$

where we have used the fact that $f_{t,r}(\mathbf{w})$ is $(\gamma+\kappa)$-strongly convex in the last inequality. Setting $\left(\frac{3}{4}\right)^{2K} \frac{\beta+\gamma+\kappa}{\gamma+\kappa} = \eta$, and noting $\mathbf{x}_r = \mathbf{z}_K$, we obtain the desired iteration complexity.

□

At the outer iteration t of Algorithm 2, we are trying to approximately minimize the objective (3.16) by iteratively (approximately) solving R instances of the "augmented" problem (3.15). Let $\mathbf{w}_t^*$ be the exact minimizer of the "large minibatch" subproblem (3.16):

$$\mathbf{w}_t^* = \underset{\mathbf{w}}{\operatorname{argmin}}\ \widetilde{f}_t(\mathbf{w}).$$

The following lemma characterizes the accelerated convergence rate.

Lemma 9 (Acceleration by universal catalyst, Theorem 3.1 of Lin et al. [2015]). *Assume that for all $r \ge 1$, we have*

$$\bar{f}_{t,r}(\mathbf{x}_r) - \bar{f}_{t,r}(\mathbf{x}_r^*) \le \frac{2}{9}\left(1 - \frac{9}{10}\sqrt{\frac{\gamma}{\gamma+\kappa}}\right)^R \cdot \left(\widetilde{f}_t(\mathbf{x}_0) - \widetilde{f}_t(\mathbf{w}_t^*)\right),$$

then

$$\widetilde{f}_t(\mathbf{x}_R) - \widetilde{f}_t(\mathbf{w}_t^*) \le \frac{800(\gamma+\kappa)}{\gamma}\left(1 - \frac{9}{10}\sqrt{\frac{\gamma}{\gamma+\kappa}}\right)^{R+1}\left(\widetilde{f}_t(\mathbf{x}_0) - \widetilde{f}_t(\mathbf{w}_t^*)\right).$$

Proof of Theorem 21

Proof. First of all, because $R = 1$, our algorithm collapses into two nested loops.

On the one hand, as we choose γ as Theorem 19 suggested, we just need to verify the inexactness conditions in Theorem 19 is satisfied, i.e., for $t = 1, \ldots, T$, we require (recall that $\mathbf{w}_t^* = \operatorname{argmin}_{\mathbf{w}} \ \widetilde{f}_t(\mathbf{w})$)

$$\widetilde{f}_t(\mathbf{w}_t) - \widetilde{f}_t(\mathbf{w}_t^*) \leq \frac{1}{10^4} \cdot \min\left(\left(\frac{T}{bm}\right)^{1/2}, \left(\frac{T}{bm}\right)^{3/2} \right) \cdot \frac{2LB}{t^3}.$$

On the other hand, we can bound the initial suboptimality $\widetilde{f}_t(\mathbf{w})$ (cf. derivation for (3.40)):

$$\widetilde{f}_t(\widetilde{\mathbf{w}}_{t-1}) - \widetilde{f}_t(\mathbf{w}_t^*) \leq L^2/\gamma.$$

Using Lemma 8, we know as long as the inequality (3.35) is satisfied, we have the desired suboptimality in $\widetilde{f}_t(\mathbf{w})$ using (cf. the derivation for (3.41))

$$K = \mathcal{O}\left(\log n(\varepsilon)\right)$$

rounds of communication, where we have plugged in the value of γ in the second step.

It remains to verify the condition (3.35), by our choice of γ and b, we have

$$b\gamma^2 = \frac{8n(\varepsilon)L^2}{bm^2B^2} \geq \frac{8n(\varepsilon)L^2}{b^*m^2B^2} = 256\beta^2 \log(md), \tag{3.39}$$

as desired.

Next we summarize the communication, computation, and memory efficiency.

Communication: the total rounds of communication required by Algorithm 2 is

$$KRT = \mathcal{O}\left(\frac{n(\varepsilon)}{mb} \log n(\varepsilon)\right).$$

Computation: For each communication round, we need to solve the local problem (3.12) using prox-SVRG. Now, in view of (3.39), we have $\beta = \mathcal{O}(\sqrt{b}\gamma)$. This implies that $\frac{\beta}{\gamma} = \mathcal{O}(\sqrt{b})$ and thus by Lemma 7, the dominant term of the iteration complexity of prox-SVRG is

$$\mathcal{O}\left(b \log \frac{\beta+\gamma}{\gamma}\right) = \mathcal{O}\left(b \log n(\varepsilon)\right).$$

Multiplying this with the number of communication rounds yields the desired computation complexity.

Memory: It is straightforward to see each machine only need to maintain b samples. □

Proof of Theorem 3

Proof. First, it is straightforward to verify the condition (3.35):

$$b(\gamma+\kappa)^2 = 256\beta^2 \log(dm).$$

Similarly to Theorem 21, we need the ratio between final versus initial error for the R AIDE iterations to be

$$\text{ratio} = \mathcal{O}(n(\varepsilon)).$$

Equating this ratio to be $\frac{800(\gamma+\kappa)}{\gamma}\left(1-\frac{9}{10}\sqrt{\frac{\gamma}{\gamma+\kappa}}\right)^{R+1}$, we have

$$\begin{aligned} R &= \frac{10}{9}\sqrt{\frac{\gamma+\kappa}{\gamma}} \log\left(\frac{800(\gamma+\kappa)}{\gamma}\cdot\frac{1}{ratio}\right) \\ &= \mathcal{O}\left(\frac{b^{1/4}m^{1/2}\cdot\beta^{1/2}B^{1/2}}{n(\varepsilon)^{1/4}\cdot L^{1/2}} \log n(\varepsilon)\right). \end{aligned}$$

Now according to Lemma 9, the final suboptimality for $\bar{f}_{t,r}(\mathbf{w})$ need to be

$$\epsilon_{\text{final}} = \frac{2}{9}\left(1 - \frac{9}{10}\sqrt{\frac{\gamma}{\gamma+\kappa}}\right)^R \cdot \left(\widetilde{f}_t(\mathbf{x}_0) - \widetilde{f}_t(\mathbf{w}_t^*)\right).$$

Let us initialize $\min_{\mathbf{w}} \bar{f}_{t,r}(\mathbf{w})$ by $\mathbf{x}_0$. By definition, we have $\bar{f}_{t,r}(\mathbf{w}) \geq \widetilde{f}_t(\mathbf{w})$ and thus

$$\begin{aligned}\epsilon_{\text{init}} &= \bar{f}_{t,r}(\mathbf{x}_0) - \bar{f}_{t,r}(\mathbf{x}_r^*) \\ &\leq \widetilde{f}_t(\mathbf{x}_0) - \widetilde{f}_t(\mathbf{x}_r*) \\ &\leq \widetilde{f}_t(\mathbf{x}_0) - \widetilde{f}_t(\mathbf{w}_t^*)\end{aligned}$$

where we have used the fact that $\mathbf{w}_t^*$ is the minimizer of $\widetilde{f}_t(\mathbf{w})$ in the second inequality.

This means we only need the initial versus final suboptimality of solving $\bar{f}_{t,r}(\mathbf{w})$ to be

$$\frac{1}{\eta} = \frac{\epsilon_{\text{init}}}{\epsilon_{\text{final}}} = \frac{9}{2}\left(1 - \frac{9}{10}\sqrt{\frac{\gamma}{\gamma+\kappa}}\right)^{-R},$$

which, according to Lemma 8, is achieved by inexact DANE with

$$\begin{aligned}K &= \mathcal{O}\left(\log\frac{1}{\eta} + \log\frac{\beta+\gamma+\kappa}{\gamma+\kappa}\right) \\ &= \mathcal{O}\left(R\sqrt{\frac{\gamma}{\gamma+\kappa}}\right) \\ &= \mathcal{O}\left(\log n(\varepsilon)\right).\end{aligned}$$

iterations.

Next we analyze the communication and computation efficiency of our algorithm.

Communication: The total rounds of communication is

$$KRT = \mathcal{O}\left(\log n(\varepsilon) \cdot \frac{b^{1/4} m^{1/2} \cdot \beta^{1/2} B^{1/2}}{n(\varepsilon)^{1/4} \cdot L^{1/2}} \log n(\varepsilon) \cdot \frac{n(\varepsilon)}{bm}\right)$$
$$= \mathcal{O}\left(\frac{n(\varepsilon)^{3/4} \cdot \beta^{1/2} B^{1/2}}{b^{3/4} m^{1/2} \cdot L^{1/2}} \log^2 n(\varepsilon)\right).$$

Computation: Similar to the case of $b \leq b^*$, for each DANE local subproblem (3.12), the sample size b is larger than its condition number. Therefore, the total computational cost is

$$\mathcal{O}(bKRT) = \mathcal{O}\left(\frac{b^{1/4} n(\varepsilon)^{3/4} \cdot \beta^{1/2} B^{1/2}}{m^{1/2} \cdot L^{1/2}} \log^2 n(\varepsilon)\right).$$

□

3.7.11 Proof of Theorem 22

Proof. On the one hand, as we choose γ as Theorem 19 suggested, we just need to verify that the inexactness conditions in Theorem 19 is satisfied, i.e., for $t = 1, \ldots, T$, we require (recall that $\mathbf{w}_t^* = \operatorname{argmin}_{\mathbf{w}} \ \widetilde{f}_t(\mathbf{w})$)

$$\widetilde{f}_t(\mathbf{w}_t) - \widetilde{f}_t(\mathbf{w}_t^*) \leq \frac{1}{10^4} \cdot \min\left(\left(\frac{T}{bm}\right)^{1/2}, \left(\frac{T}{bm}\right)^{3/2}\right) \cdot \frac{LB}{t^3}.$$

On the other hand, we can bound the initial suboptimality of $\widetilde{f}_t(\mathbf{w})$ when initializing from $\mathbf{w}_{t-1}$. This is because, by the optimality of $\mathbf{w}_t^*$, we have $||\mathbf{w}_t^* - \mathbf{w}_{t-1}|| = ||\frac{1}{\gamma}\nabla\phi_{I_t}(\mathbf{w}_t^*)|| \leq L/\gamma$, and

$$\widetilde{f}_t(\mathbf{w}_{t-1}) - \widetilde{f}_t(\mathbf{w}_t^*) = 0 + \phi_{I_t}(\mathbf{w}_{t-1}) - \frac{\gamma}{2}||\mathbf{w}_t^* - \mathbf{w}_{t-1}||^2 - \phi_{I_t}(\mathbf{w}_t^*)$$
$$\leq \phi_{I_t}(\mathbf{w}_{t-1}) - \phi_{I_t}(\mathbf{w}_t^*) \leq L||\mathbf{w}_t^* - \mathbf{w}_{t-1}|| \leq L^2/\gamma. \tag{3.40}$$

Combining the above two inequalities, the initial versus final error for the K DSVRG iterations is bounded by

$$\begin{aligned}
&10^4 \cdot \max\left(\left(\frac{bm}{T}\right)^{1/2}, \left(\frac{bm}{T}\right)^{3/2}\right) \cdot t^3 \cdot \frac{L}{B\gamma} \\
&= 10^4 \cdot \max\left(\left(\frac{bm}{T}\right)^{1/2}, \left(\frac{bm}{T}\right)^{3/2}\right) \cdot T^3 \cdot \frac{L}{B} \cdot \frac{bmB}{\sqrt{8n(\varepsilon)}L} \\
&= \mathcal{O}\left(\max\left(\frac{n(\varepsilon)^2}{bm}, bm \cdot n(\varepsilon)\right)\right) \\
&= \mathcal{O}\left(n^2(\varepsilon)\right)
\end{aligned}$$

where we have used the definition of γ and $T = \frac{n(\varepsilon)}{bm}$ in the first and second step respectively.

By the iteration complexity results for sampling without-replacement DSVRG [Shamir, 2016, Theorem 4], we have the desired suboptimality in $\widetilde{f}_t(\mathbf{w})$ using

$$K = \mathcal{O}\left(\log n(\varepsilon)\right) \tag{3.41}$$

iterations, as long as the batch size b/p_i is larger than the problem condition number.

Now, the condition number of $\widetilde{f}(\mathbf{w})$ is

$$\frac{\beta + \gamma}{\gamma} = \mathcal{O}\left(\frac{\beta bmB}{\sqrt{n(\varepsilon)}L}\right).$$

Equating this to the batch size b/p_i yields the p_i specified in the theorem. It is also easy to check that $\frac{K}{\gamma} = \mathcal{O}(bm)$, i.e., the total number of stochastic updates is less than the total number of samples, as required by Shamir [2016, Theorem 4].

Communication: the total rounds of communication required by Algorithm 3 is

$$KT = \mathcal{O}\left(\frac{n(\varepsilon)}{mb}\log n(\varepsilon)\right).$$

Computation: For each communication round, each machine need to compute the local full gradient, which can be done in parallel, and then one of the machines perform b/p_i steps of stochastic update. So the computation cost is

$$KT\left(b+\frac{b}{p_i}\right)=\mathcal{O}\left(\frac{n(\varepsilon)}{m}\log n(\varepsilon)\right).$$

Memory: It is straightforward to see each machine only need to maintain b samples. $\square$

CHAPTER 4

MORE EFCIENT DISTRIBUTED LEARNING VIA SKETCHING

4.1 Motivation and problem set-up

Machine learning is nowadays successfully applied to massive data sets collected from various domains. One of the major challenges in applying machine learning methods to massive data sets is how to effectively utilize available computational resources when building predictive and inferential models, while utilizing data in a statistically optimal way. One approach to tackling massive data sets is via building distributed computer systems and developing distributed learning algorithms. However, distributed systems may not always be available. Furthermore, the cost of running a distributed system can be much higher than one can afford, making distributed learning unsuitable for all scenarios. An alternative approach is to use the state-of-the-art randomized optimization algorithms to accelerate the training process. For example, many optimization algorithms are available for solving regularized empirical risk minimization problems, with provably fast convergence and low computational cost per iteration (see [Johnson and Zhang, 2013, Zhang et al., 2013a, Defazio et al., 2014] for examples). It is worth pointing out at this point that the speed of these optimization methods still heavily depends on the condition number of the problem at hand, which is undesirable for many real world problems.

Sketching has emerged as a technique for big data analytics [Woodruff et al., 2014]. The idea behind sketching is to approximate the solution of the original problem by solving a sketched, smaller scale problem. For example, sketching has been used to *approximately* solve various large-scale problems, ranging from least square regression and robust regression to low-rank approximation and singular value decomposition (see [Halko et al., 2011, Mahoney et al., 2011, Lu et al., 2013, Alaoui and Mahoney, 2015, Woodruff et al., 2014, Raskutti and

Mahoney, 2015, Yang et al., 2015, Oymak et al., 2015, Oymak and Tropp, 2015, Drineas and Mahoney, 2016] and references therein), and has been implemented in high-quality software packages of least-square packages [Avron et al., 2010, Meng et al., 2014]. However, one major drawback of sketching is that it is typically not suitable in scenarios where a *highly accurate* solution is needed. To obtain a solution with exponentially smaller approximation error, we often also need to increase the sketching dimension *exponentially* as well.

Recent work on "iterative sketch", iterative Hessian sketch (IHS) [Pilanci and Wainwright, 2016] and iterative dual random projection (IDRP) [Zhang et al., 2014], has improved the situation. These methods are able to refine the accuracy of their solution by iteratively solving small scale sketched problem. Hessian sketch [Pilanci and Wainwright, 2016] is designed to reduce the sample size of the original problem, while dual random projection [Zhang et al., 2014] is proposed to reduce the dimensionality of data. As a consequence, when the sample size and feature dimension are both large, IHS and IDRP still need to solve relatively large-scale subproblems as they can only sketch the problem from one perspective.

In this chapter, we address the problem of the *recovery* of optimal solution for big and high-dimensional data by solving small sketched problems of original problem. We make the following contributions. First, we propose an accelerated version of IHS that is computationally as effective as IHS at each iteration, but requires provably fewer number of sketching iterations to reach a certain accuracy. Next, we reveal a primal-dual connection between IHS [Pilanci and Wainwright, 2016] and IDRP [Zhang et al., 2014], that were independently proposed by two different groups of researchers. We show that these two methods are equivalent in the sense that the dual random projection is essentially performing the Hessian sketch in the dual space. This connection allows us to provide a unified analysis of IHS and IDRP, and also develop an accelerated sketching schema. Finally, we alleviate the computational issues raised by big and high-dimensional learning problems. We propose a *primal-dual sketching* method that can simultaneously reduce the sample size and dimension of the problem, and recover the optimal solution to the original large-scale high-dimensional problem with provable convergence guarantees. We also demonstrate applicability of the iterative sketching techniques for the distributed optimization problems where the data are partitioned across machines, either by samples or features.

Chapter Organization. The rest of this chapter is organized as follows: in Section 4.2.1 we review the iterative Hessian sketch as an optimization process and propose a new algorithm with faster convergence rate. In Section 4.2.3 we show that the dual random projection is equivalent to the Hessian sketch. This equivalence allows us to propose the corresponding accelerated dual random projection. In Section 4.2.6 we combine the sketching from both primal and dual perspectives, and propose an iterative algorithm that reduces both sample size and problem dimension. Theoretical properties of are investigated in Section 4.3, while technical details are deferred to Appendix. In Section 4.4 we discuss an application of the iterative sketching for distributed optimization. We present experiments in Section 7.5 to support our theoretical results.

Notation. We use bold-faced letters, such as $\mathbf{w}$, to denote vectors, and bold-faced capital letters, such as $\mathbf{X}$, to denote matrices. The set of real numbers is denoted by $\mathbb{R}$. Given a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, we define the following matrix induced norm for any vector $\mathbf{w} \in \mathbb{R}^p$,

$$||\mathbf{w}||_{\mathbf{X}} = \sqrt{\frac{\mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w}}{n}}.$$

We use $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to denote the multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. We use $\mathbf{I}_n$ to denote the identity matrix of size $n \times n$. The maximum and minimum eigenvalues of $\mathbf{H}$ are $\lambda_{\max}(\mathbf{H})$ and $\lambda_{\min}(\mathbf{H})$, respectively. The condition number of a matrix $\mathbf{H}$ is denoted by $\kappa(\mathbf{H})$, which is the ratio of the largest to smallest singular value in the singular value decomposition of $\mathbf{H}$. For two sequences $\{a_n\}_{n=1}^{\infty}$ and $\{a_n\}_{n=1}^{\infty}$, we denote

$a_n \lesssim b_n$ if $a_n \leq C b_n$ always holds for n large enough with some constant C, and denote $a_n \gtrsim b_n$ if $b_n \lesssim a_n$. We also use the notation $a_n = \mathcal{O}(b_n)$ if $a_n \lesssim b_n$, and use $\widetilde{\mathcal{O}}(\cdot)$ for $\mathcal{O}(\cdot)$ to hide logarithmic factors.

4.2 Insights on iterative sketching algorithms on single machines

In this section, we describe our new insights on single machine iterative sketching algorithms from an optimization view.

4.2.1 Iterative Hessian sketch as optimization with preconditioning

We first review the iterative Hessian sketch proposed in [Pilanci and Wainwright, 2016]. We present the iterative Hessian sketch as an iterative preconditioned optimization process. This allows us to propose a faster iterative algorithm by solving a different sketched problem.

Consider the following ℓ_2 regularized least-squares problem, also known as the ridge regression:

$$\min_{\mathbf{w}\in\mathbb{R}^p} P(\mathbf{X},\mathbf{y};\mathbf{w}) = \min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2. \tag{4.1}$$

where $\mathbf{X} \in \mathbb{R}^{n\times p}$ is the data matrix, $\mathbf{y} \in \mathbb{R}^n$ is the response vector, and λ is the tuning parameter. Let $\mathbf{w}^*$ denote the optimum of problem (4.1) which can be computed in a closed form as

$$\mathbf{w}^* = \left(\lambda \mathbf{I}_p + \frac{\mathbf{X}^\top\mathbf{X}}{n}\right)^{-1} \frac{\mathbf{X}^\top\mathbf{y}}{n},$$

however, to compute the closed-form solution requires one to construct and invert the covariance matrix, which can take $\mathcal{O}(np^2 + p^3)$ time to finish.

Sketching has become a widely used technique for efficiently finding an approximate solution to (4.1) when both n and p are large [Drineas et al., 2011, Mahoney et al., 2011, Woodruff et al., 2014]. To avoid solving a problem of huge sample size, the traditional sketching techniques (for example, [Sarlos, 2006, Pilanci and Wainwright, 2015]) reduce the sample size from n to m, with $m \ll n$, and solve the following sketched ℓ_2 regularized least-squares problem:

$$\min_{\mathbf{w}\in\mathbb{R}^p} P(\boldsymbol{\Pi}^\top \mathbf{X}, \boldsymbol{\Pi}^\top \mathbf{y}; \mathbf{w}) = \min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\boldsymbol{\Pi}^\top \mathbf{y} - \boldsymbol{\Pi}^\top \mathbf{X}\mathbf{w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2, \tag{4.2}$$

where $\boldsymbol{\Pi} \in \mathbb{R}^{n\times m}$ is a sketching matrix. The problem (4.2) can be solved faster and with less storage as long as we can choose $m \ll n$. Typical choice of $\boldsymbol{\Pi}$ includes a random matrix with Gaussian or Rademacher entries, sub-sampled randomized Hadamard transform [Boutsidis and Gittens, 2013], and sub-sampled Randomized Fourier Transform [Rokhlin and Tygert, 2008]. See discussions in Section 2.1 of [Pilanci and Wainwright, 2016] for more details.

Though the classical sketching has been successful in various problems and has provable guarantees, as shown in [Pilanci and Wainwright, 2016], there is an approximation precision limit for classical sketching methods could achieve, given a fixed sketching dimension. To obtain an approximate solution with high precision, the sketching dimension m often needs to be of the same order as n. This is impractical as the goal of sketching is to speed up the algorithms via reducing the sample size.

The main idea behind the Hessian sketch [Pilanci and Wainwright, 2016] is based on the following equivalent formulation of (4.1):

$$\min_{\mathbf{w}\in\mathbb{R}^p} P(\mathbf{X}, \mathbf{y}; \mathbf{w}) = \min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y}||_2^2 + \frac{1}{2n}||\mathbf{X}\mathbf{w}||_2^2 - \frac{1}{n}\langle \mathbf{y}, \mathbf{X}\mathbf{w}\rangle + \frac{\lambda}{2}||\mathbf{w}||_2^2. \tag{4.3}$$

In the Hessian sketch one only sketches the quadratic part $||\mathbf{X}\mathbf{w}||_2^2$ with respect to $\mathbf{X}$, but not the linear part $\langle \mathbf{y}, \mathbf{X}\mathbf{w}\rangle$, leading to the following problem:

$$\min_{\mathbf{w}\in\mathbb{R}^p} P_{\text{HS}}(\mathbf{X}, \mathbf{y}; \boldsymbol{\Pi}, \mathbf{w}) = \min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y}||_2^2 + \frac{1}{2n}||\boldsymbol{\Pi}^\top \mathbf{X}\mathbf{w}||_2^2 - \frac{1}{n}\langle \mathbf{y}, \mathbf{X}\mathbf{w}\rangle + \frac{\lambda}{2}||\mathbf{w}||_2^2. \tag{4.4}$$

The solution to the problem (4.4) has the following closed form solution:

$$\widehat{\mathbf{w}}_{\text{HS}} = \left(\lambda \mathbf{I}_p + \frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{n}\right)^{-1} \frac{\mathbf{X}^\top \mathbf{y}}{n}. \tag{4.5}$$

Compared to the classical sketch where both the data matrix $\mathbf{X}$ and the response vector $\mathbf{y}$ are sketched, in the Hessian sketch one only sketches the Hessian matrix, through the following transform:

$$\mathbf{X}^\top \mathbf{X} \to \mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}.$$

The Hessian sketch suffers from the same approximation limit as the classical sketch. However, one notable feature of the Hessian sketch is that one can implement an iterative extension to refine the accuracy of the approximation. Define the initial Hessian sketch approximation as $\widehat{\mathbf{w}}_{\text{HS}}^{(1)}$:

$$\widehat{\mathbf{w}}_{\text{HS}}^{(1)} = \arg\min_{\mathbf{w}} \mathbf{w}^\top \left(\frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{2n} + \frac{\lambda}{2}\mathbf{I}_p\right) \mathbf{w} - \frac{1}{n}\langle \mathbf{y}, \mathbf{X}\mathbf{w}\rangle.$$

A refinement of $\widehat{\mathbf{w}}_{\text{HS}}^{(1)}$ can be obtained by considering the following optimization problem

$$\begin{aligned}
&\arg\min_{\mathbf{u}} \frac{1}{2n} ||\mathbf{y} - \mathbf{X}(\mathbf{u} + \widehat{\mathbf{w}}_{\text{HS}}^{(1)})||_2^2 + \frac{\lambda}{2}||(\mathbf{u} + \widehat{\mathbf{w}}_{\text{HS}}^{(1)})||_2^2 \\
&= \arg\min_{\mathbf{u}} \mathbf{u}^\top \left(\frac{\mathbf{X}^\top \mathbf{X}}{2n} + \frac{\lambda}{2}\mathbf{I}_p\right) \mathbf{u} - \left\langle \frac{\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}_{\text{HS}}^{(t)})}{n} - \lambda \widehat{\mathbf{w}}_{\text{HS}}^{(t)}, \mathbf{u} \right\rangle,
\end{aligned}$$

whose optimum is $\mathbf{w}^* - \widehat{\mathbf{w}}_{\text{HS}}^{(1)}$. The main idea of the iterative Hessian sketch is to approximate the residual solution $\mathbf{w}^* - \widehat{\mathbf{w}}_{\text{HS}}^{(1)}$ by the Hessian sketch. At iteration t, $\mathbf{w}^* - \widehat{\mathbf{w}}_{\text{HS}}^{(t)}$ is approximated by $\widehat{\mathbf{u}}^{(t)}$ that minimizes the following problem

$$\widehat{\mathbf{u}}^{(t)} = \arg\min_{\mathbf{u}} \mathbf{u}^\top \left(\frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{2n} + \frac{\lambda}{2}\mathbf{I}_p\right) \mathbf{u} - \left\langle \frac{\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}_{\text{HS}}^{(t)})}{n} - \lambda \widehat{\mathbf{w}}_{\text{HS}}^{(t)}, \mathbf{u} \right\rangle, \tag{4.6}$$

Algorithm 4 Iterative Hessian Sketch (IHS).

```
Input: Data $\mathbf{X}, \mathbf{y}$, sketching matrix $\mathbf{\Pi}$.
Initialization: $\widehat{\mathbf{w}}_{\text{HS}}^{(0)} = \mathbf{0}$.
for $t = 0, 1, 2, \ldots$ do
    Update the approximation by $\widehat{\mathbf{w}}_{\text{HS}}^{(t+1)} = \widehat{\mathbf{w}}_{\text{HS}}^{(t)} + \widehat{\mathbf{u}}^{(t)}$, where $\widehat{\mathbf{u}}^{(t)}$ is obtained by solving
    the sketched problem (4.6).
end
```

and the new approximation $\widehat{\mathbf{w}}_{\text{HS}}^{(t+1)}$ is updated as

$$\widehat{\mathbf{w}}_{\text{HS}}^{(t+1)} = \widehat{\mathbf{w}}_{\text{HS}}^{(t)} + \widehat{\mathbf{u}}^{(t)}.$$

The algorithm for IHS is shown in Algorithm 4. Since (4.6) is a sketched problem with sample size m, it can be solved more efficiently than the original problem (4.1). Notice that we can reuse the previously sketched data matrix $\mathbf{\Pi}^\top \mathbf{X}$ without constructing any new random sketching matrices. [Pilanci and Wainwright, 2016] showed that the approximation error of IHS is exponentially decreasing with the number of sketching iterations. Thus IHS can find an approximate solution with an ϵ-approximation error within $\mathcal{O}(\log(1/\epsilon))$ iterations, as long as the sketching dimension m is large enough. IHS was originally developed for the least-squares problem in (4.1), the idea can be extended to solve more general problems, such as constrained least-squares [Pilanci and Wainwright, 2016], optimization with self-concordant loss [Pilanci and Wainwright, 2017], as well as non-parametric methods [Yang et al., 2017].

Though IHS improved the classical sketching by enabling us to find a high quality approximation more efficiently, it is imperfect due to the following two reasons. First, the guarantee that the approximation error decreases exponentially for IHS relies on the sketching dimension being large enough. The necessary sketching dimension depends on the intrinsic complexity of the problem, and, if the sketching dimension is too small, IHS can diverge, obtaining arbitrary worse approximation. Second, even when the sketching dimension is large enough, the speed at which the approximation error decreases in IHS can be significantly improved.

Now, we show that the iterative Hessian sketch is in fact an optimization process with *preconditioning*. This view allows us to develop better iterative algorithms by searching the conjugate directions. For notation simplicity, let

$$\mathbf{H} = \frac{\mathbf{X}^\top \mathbf{X}}{n} + \lambda \mathbf{I}_p \quad \text{and} \quad \widetilde{\mathbf{H}} = \frac{\mathbf{X}^\top \mathbf{\Pi} \mathbf{\Pi}^\top \mathbf{X}}{n} + \lambda \mathbf{I}_p.$$

Let

$$\nabla P(\mathbf{w}) = -\frac{\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\mathbf{w})}{n} + \lambda \mathbf{w}$$

denote the gradient of $P(\mathbf{X}, \mathbf{y}; \mathbf{w})$ with respect to $\mathbf{w}$. Then the IHS algorithm can be seen as performing the following iterative update

$$\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} = \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)} - \widetilde{\mathbf{H}}^{-1} \nabla P(\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)}),$$

which is like a Newton update where we replace the true Hessian $\mathbf{H}$ with the sketched Hessian $\widetilde{\mathbf{H}}$. Another way to think about this update is via the change of variable $\mathbf{z} = \widetilde{\mathbf{H}}^{1/2}\mathbf{w}$ and then applying the gradient descent in the $\mathbf{z}$ space

$$\widehat{\mathbf{z}}^{(t+1)} = \widehat{\mathbf{z}}^{(t)} - \nabla_{\mathbf{z}} P(\widetilde{\mathbf{H}}^{-1/2} \widehat{\mathbf{z}}^{(t)}) = \widehat{\mathbf{z}}^{(t)} - \widetilde{\mathbf{H}}^{-1/2} \nabla_{\mathbf{x}} P(\widetilde{\mathbf{H}}^{-1/2} \widehat{\mathbf{z}}^{(t)}).$$

Multiplying by $\widetilde{\mathbf{H}}^{-1/2}$, changes the update back to the original space, leading back to the IHS update

$$\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} = \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)} - \widetilde{\mathbf{H}}^{-1} \nabla P(\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)}).$$

With above discussion, we see that the iterative Hessian sketch is in fact an optimization process with the sketched Hessian as preconditioning.

4.2.2 Accelerated IHS via preconditioned conjugate gradient

In this section, we present the accelerated iterative Hessian sketch (Acc-IHS) algorithm by utilizing the idea of preconditioned conjugate gradient. Conjugate gradient is known to have better convergence properties than gradient descent in solving linear systems [Hestenes and Stiefel, 1952, Nocedal and Wright, 2006]. Since the iterative Hessian sketch is performing the gradient descent (with stepsize 1) in the transformed space $\mathbf{z} = \widetilde{\mathbf{H}}^{1/2}\mathbf{w}$, it can be accelerated by performing the conjugate gradient descent instead. Equivalently, we can implicitly transform the space by defining inner product as $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \widetilde{\mathbf{H}} \mathbf{y}$.

This leads to the algorithm Acc-IHS as detailed in Algorithm 5. At each iteration, the solver is called for the following sketched linear system:

$$\widehat{\mathbf{u}}^{(t)} = \arg\min_{\mathbf{u}} \mathbf{u}^\top \left(\frac{\mathbf{X}^\top \mathbf{\Pi} \mathbf{\Pi}^\top \mathbf{X}}{2n} + \frac{\lambda}{2} \mathbf{I}_p \right) \mathbf{u} - \left\langle \mathbf{r}^{(t)}, \mathbf{u} \right\rangle. \tag{4.7}$$

Unlike IHS, which uses $\widetilde{\mathbf{H}}^{-1} \nabla P(\widehat{\mathbf{w}}_{\text{HS}}^{(t)})$ as the update direction at iteration t, Acc-IHS uses $\mathbf{p}^{(t)}$ as the update direction where $\mathbf{p}^{(t)}$ is chosen to satisfy the conjugate condition: $\forall t_1, t_2 \geq 0, t_1 \neq t_2$

$$\left(\mathbf{p}^{(t_1)}\right)^\top \widetilde{\mathbf{H}}^{-1/2} \mathbf{H} \widetilde{\mathbf{H}}^{-1/2} \mathbf{p}^{(t_2)} = 0.$$

Since the updating direction is conjugate to the previous directions, it is guaranteed that after p iterations we reach the exact minimizer, that is,

$$\widehat{\mathbf{w}}_{\text{HS}}^{(t)} = \mathbf{w}^*, \quad \forall t \geq p.$$

Moreover, Acc-IHS has the same computational cost as the standard IHS in solving each sketched sub-problem. However, the convergence rate of Algorithm 5 is much faster than

Algorithm 5 Accelerated Iterative Hessian Sketch (Acc-IHS).

```
Input: Data $\mathbf{X}, \mathbf{y}$, sketching matrix $\mathbf{\Pi}$.
Initialization: $\widehat{\mathbf{w}}_{\mathrm{HS}}^{(0)} = \mathbf{0}, \mathbf{r}^{(0)} = -\frac{\mathbf{X}^\top \mathbf{y}}{n}$.
Compute $\widehat{\mathbf{u}}^{(0)}$ by solving (4.7), and update $\mathbf{p}^{(0)} = -\widehat{\mathbf{u}}^{(0)}$, calculate $\mathbf{v}^{(0)} = \left(\frac{\mathbf{X}^\top \mathbf{X}}{n} + \lambda \mathbf{I}_p\right) \mathbf{p}^{(0)}$.
for $t = 0, 1, 2, \ldots$ do
    Calculate $\alpha^{(t)} = \frac{\langle \mathbf{r}^{(t)}, \mathbf{u}^{(t)} \rangle}{\langle \mathbf{p}^{(t)}, \mathbf{v}^{(t)} \rangle}$
    Update the approximation by $\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} = \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)} + \alpha^{(t)} \mathbf{p}^{(t)}$.
    Update $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} + \alpha^{(t)} \mathbf{v}^{(t)}$.
    Update $\mathbf{u}^{(t+1)}$ by solving (4.7).
    Update $\beta^{(t+1)} = \frac{\langle \mathbf{r}^{(t+1)}, \mathbf{u}^{(t)} \rangle}{\langle \mathbf{r}^{(t)}, \mathbf{r}^{(t)} \rangle}$.
    Update $\mathbf{p}^{(t+1)} = -\mathbf{u}^{(t+1)} + \beta^{(t+1)} \mathbf{p}^{(t)}$.
    Update $\mathbf{v}^{(t+1)} = \left(\frac{\mathbf{X}^\top \mathbf{X}}{n} + \lambda \mathbf{I}_p\right) \mathbf{p}^{(t+1)}$.
end
```

IHS, that is, it requires solving much smaller number of sketched sub-problems compared to IHS to reach the same approximation accuracy.

4.2.3 Equivalence between dual random projection and Hessian sketch

While Hessian sketch [Pilanci and Wainwright, 2016] tries to resolve the issue of huge sample size, Dual Random Projection [Zhang et al., 2013b, 2014] is aimed at resolving the issue of high-dimensionality by using random projections as a tool for reducing the dimension of data. Again, we consider the standard ridge regression problem in (4.1). A random projection is now used to transform the original problem (4.1) to a low-dimensional problem:

$$\min_{\mathbf{z} \in \mathbb{R}^p} P_{\mathrm{RP}}(\mathbf{XR}, \mathbf{y}; \mathbf{z}) = \min_{\mathbf{z} \in \mathbb{R}^d} \frac{1}{2n} ||\mathbf{y} - \mathbf{XRz}||_2^2 + \frac{\lambda}{2} ||\mathbf{z}||_2^2, \tag{4.8}$$

where $\mathbf{R} \in \mathbb{R}^{p \times d}$ is a random projection matrix, and $d \ll p$.

Let $\widehat{\mathbf{z}} = \arg\min_{\mathbf{z}} P_{\mathrm{RP}}(\mathbf{XR}, \mathbf{y}; \mathbf{z})$. If we want to recover the original high-dimensional solution, [Zhang et al., 2014] observed that the naive solution $\widehat{\mathbf{w}}_{\mathrm{RP}} = \mathbf{R}\widehat{\mathbf{z}}$ results in a bad approximation. Instead, the optimal solution of the original problem, $\mathbf{w}^*$, is recovered from the dual solution, leading to the dual random projection (DRP) approach that we explain below. The dual problem of the optimization problem in (4.1) is

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} D(\mathbf{X},\mathbf{y};\boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} -\frac{1}{2n}\boldsymbol{\alpha}^\top\boldsymbol{\alpha} + \frac{\mathbf{y}^\top\boldsymbol{\alpha}}{n} - \frac{1}{2\lambda n^2}\boldsymbol{\alpha}^\top\mathbf{X}\mathbf{X}^\top\boldsymbol{\alpha}. \tag{4.9}$$

Let $\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} D(\mathbf{X},\mathbf{y};\boldsymbol{\alpha})$ be the dual optimal solution. By the standard primal-dual theory [Boyd and Vandenberghe, 2004], we have the following connection between the optimal primal and dual solutions:

$$\boldsymbol{\alpha}^* = \mathbf{y} - \mathbf{X}\mathbf{w}^* \quad \text{and} \quad \mathbf{w}^* = \frac{1}{\lambda n}\mathbf{X}^\top\boldsymbol{\alpha}^*. \tag{4.10}$$

The dual random projection procedure works as follows. First, we construct and solve the low-dimensional, randomly projected problem (4.8) and obtain the solution $\widehat{\mathbf{z}}$. Next, we calculate the approximated dual variables by

$$\widehat{\boldsymbol{\alpha}}_{\text{DRP}} = \mathbf{y} - \mathbf{X}\mathbf{R}\widehat{\mathbf{z}}, \tag{4.11}$$

based on the approximated dual solution $\widehat{\boldsymbol{\alpha}}_{\text{DRP}}$. Finally, we recover the primal solution as:

$$\widehat{\mathbf{w}}_{\text{DRP}} = \frac{1}{\lambda n}\mathbf{X}^\top\widehat{\boldsymbol{\alpha}}_{\text{DRP}}. \tag{4.12}$$

Combining the steps above, it is easy to see that the dual random projection for ridge regression has the following closed form solution:

$$\widehat{\mathbf{w}}_{\text{DRP}} = \frac{\mathbf{X}^\top}{n}\left(\lambda\mathbf{I}_n + \frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top\mathbf{X}^\top}{n}\right)^{-1}\mathbf{y}. \tag{4.13}$$

The recovered solution from the dual, $\widehat{\mathbf{w}}_{\text{DRP}}$, has much better approximation compared to the solution recovered directly from primal problem $\widehat{\mathbf{w}}_{\mathrm{RP}}$. Specifically, $\widehat{\mathbf{w}}_{\mathrm{RP}}$ is always a poor approximation of $\mathbf{w}^*$, because $\widehat{\mathbf{w}}_{\mathrm{RP}}$ lives in a random subspace spanned by the random projection matrix $\mathbf{R}$, while $\widehat{\mathbf{w}}_{\mathrm{DRP}}$ can be a good approximation of $\mathbf{w}^*$ as long as the projected dimension d is large enough [Zhang et al., 2014]. Finally, an iterative extension of DRP can exponentially reduce the approximation error [Zhang et al., 2014], that we explain next.

Suppose at iteration t we have the approximate solution $\widehat{\mathbf{w}}_{\mathrm{DRP}}^{(t)}$. Consider the following optimization problem:

$$\min_{\mathbf{u}\in\mathbb{R}^p} \frac{1}{2n}||\mathbf{y} - \mathbf{X}(\mathbf{u} + \widehat{\mathbf{w}}_{\mathrm{DRP}}^{(t)})||_2 + \frac{\lambda}{2}||\mathbf{u} + \widehat{\mathbf{w}}_{\mathrm{DRP}}^{(t)}||_2^2, \tag{4.14}$$

with optimum at $\mathbf{w}^* - \widehat{\mathbf{w}}_{\mathrm{DRP}}^{(t)}$. Similar to the iterative Hessian sketch, the idea behind the iterative dual random projection (IDRP) is to approximate the residual $\mathbf{w}^* - \widehat{\mathbf{w}}_{\mathrm{DRP}}^{(t)}$ by applying dual random projection again. Given $\widehat{\mathbf{w}}_{\mathrm{DRP}}^{(t)}$ we construct the following randomly projected problem:

$$\min_{\mathbf{z}\in\mathbb{R}^d} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}_{\mathrm{DRP}}^{(t)} - \mathbf{X}\mathbf{R}\mathbf{z}||_2^2 + \frac{\lambda}{2}||\mathbf{z} + \mathbf{R}^\top\mathbf{w}_{\mathrm{DRP}}^{(t)}||_2^2. \tag{4.15}$$

Let $\widehat{\mathbf{z}}^{(t)}$ to be the solution of (4.15), which is used to refine the dual and primal variables as:

$$\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}^{(t+1)} = \mathbf{y} - \mathbf{X}\mathbf{w}_{\mathrm{DRP}}^{(t)} - \mathbf{X}\mathbf{R}\widehat{\mathbf{z}},$$

and

$$\widehat{\mathbf{w}}_{\mathrm{DRP}}^{(t+1)} = \frac{1}{\lambda n}\mathbf{X}^\top\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}^{(t+1)}.$$

The iterative dual random projection (IDRP) algorithm is shown in Algorithm 6. Here we presented the idea in the context of ℓ_2 regularized least-squares. However, the iterative dual random projection can be used to solve any ℓ_2 regularized empirical loss minimization problem, as long as the loss function is smooth [Zhang et al., 2014], such as, logistic regression

Algorithm 6 Iterative Dual Random Projection (IDRP).

```
Input: Data $\mathbf{X}, \mathbf{y}$, projection matrix $\mathbf{R}$.
Initialization: $\widehat{\mathbf{w}}_{\mathrm{DRP}}^{(0)} = \mathbf{0}$.
for $t = 0, 1, 2, \ldots$ do
    Solve the projected problem in (4.15) and obtain solution $\widehat{\mathbf{z}}^{(t)}$.
    Update dual approximation: $\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}^{(t+1)} = \mathbf{y} - \mathbf{X}\mathbf{w}_{\mathrm{DRP}}^{(t)} - \mathbf{X}\mathbf{R}\widehat{\mathbf{z}}^{(t)}$.
    Update primal approximation: $\widehat{\mathbf{w}}_{\mathrm{DRP}}^{(t+1)} = \frac{1}{\lambda n}\mathbf{X}^\top \widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}^{(t+1)}$.
end
```

or support vector machines with smoothed hinge loss.

The iterative dual random projection algorithm is a powerful algorithm for dealing with high-dimensional problems, but it suffers from the same limitations as the iterative Hessian sketch. First, the projection dimension needs to be large enough to guarantee that the approximation error decreases exponentially. Second, the convergence speed is not optimal. We address both of these issues next. We will show that the dual random projection is equivalent to an application of the Hessian sketch procedure on the dual problem. This connection will allow us to develop an accelerated iterative dual random projection akin to accelerated the iterative Hessian sketch algorithm presented earlier.

4.2.4 Dual random projection is Hessian sketch in dual space

We present the equivalence connection between Hessian sketch and dual random projection. Note that the Hessian sketch is used for *sample reduction*, while the dual random projection is utilized for *dimension reduction.* Recall that the dual maximization objective (4.9) is quadratic with respect to $\boldsymbol{\alpha}$. We can write it in the equivalent form as:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \boldsymbol{\alpha}^\top \left(\frac{\mathbf{X}\mathbf{X}^\top}{2\lambda n} + \frac{1}{2}\mathbf{I}_n \right) \boldsymbol{\alpha} - \langle \mathbf{y}, \boldsymbol{\alpha} \rangle. \tag{4.16}$$

By applying the Hessian sketch with sketching matrix $\mathbf{R} \in \mathbb{R}^{p \times d}$, we find an approximate solution for $\boldsymbol{\alpha}^*$ as:

$$\widehat{\boldsymbol{\alpha}}_{\text{HS}} = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \boldsymbol{\alpha}^\top \left(\frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top \mathbf{X}^\top}{2\lambda n} + \frac{1}{2}\mathbf{I}_n \right) \boldsymbol{\alpha} - \langle \mathbf{y}, \boldsymbol{\alpha} \rangle, \tag{4.17}$$

which has the closed form solution as

$$\widehat{\boldsymbol{\alpha}}_{\text{HS}} = \lambda \left(\lambda \mathbf{I}_n + \frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top \mathbf{X}^\top}{n} \right)^{-1} \mathbf{y}.$$

Substituting $\widehat{\boldsymbol{\alpha}}_{\text{HS}}$ in the primal-dual connection (4.10), gives us the following approximation to the original problem

$$\widehat{\mathbf{w}} = \frac{\mathbf{X}^\top}{n} \left(\lambda \mathbf{I}_n + \frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top \mathbf{X}^\top}{n} \right)^{-1} \mathbf{y},$$

which is the same as the DRP approximation in (4.13). From this discussion, we see that *the Dual Random Projection is the Hessian sketch applied in the dual space.* To summarize, for ridge regression problem (4.1) one has closed form solutions for various sketching techniques as:

$$
\begin{aligned}
\textbf{Original}: \quad \mathbf{w}^* &= \left(\lambda \mathbf{I}_p + \frac{\mathbf{X}^\top \mathbf{X}}{n}\right)^{-1} \frac{\mathbf{X}^\top \mathbf{y}}{n} \\
&= \frac{\mathbf{X}^\top}{n}\left(\lambda \mathbf{I}_n + \frac{\mathbf{X}\mathbf{X}^\top}{n}\right)^{-1} \mathbf{y}; \\
\textbf{Classical Sketch}: \quad \widehat{\mathbf{w}}_{\mathrm{CS}} &= \left(\lambda \mathbf{I}_p + \frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{n}\right)^{-1} \frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{y}}{n}; \\
\textbf{Random Projection}: \quad \widehat{\mathbf{w}}_{\mathrm{RP}} &= \mathbf{R}\left(\lambda \mathbf{I}_d + \frac{\mathbf{R}^\top \mathbf{X}^\top \mathbf{X}\mathbf{R}}{n}\right)^{-1} \mathbf{R}^\top \frac{\mathbf{X}^\top \mathbf{y}}{n}; \\
\textbf{Hessian Sketch}: \quad \widehat{\mathbf{w}}_{\mathrm{HS}} &= \left(\lambda \mathbf{I}_p + \frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{n}\right)^{-1} \frac{\mathbf{X}^\top \mathbf{y}}{n}; \\
\textbf{Dual Random Projection}: \quad \widehat{\mathbf{w}}_{\mathrm{DRP}} &= \frac{\mathbf{X}^\top}{n}\left(\lambda \mathbf{I}_n + \frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top \mathbf{X}^\top}{n}\right)^{-1} \mathbf{y}.
\end{aligned}
$$

As we can see above, the Hessian sketch is sketching the *covariance matrix*:

$$\mathbf{X}^\top \mathbf{X} \to \mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X},$$

while DRP is sketching the *Gram matrix*:

$$\mathbf{X}\mathbf{X}^\top \to \mathbf{X}\mathbf{R}\mathbf{R}^\top \mathbf{X}^\top.$$

4.2.5 Accelerated iterative dual random projection

Based on the equivalence between dual random projection and Hessian sketch established in Section 4.2.4, we propose an accelerated iterative dual random projection algorithm, which improves the convergence speed of standard iterative DRP procedure [Zhang et al., 2014]. The algorithm is shown in Algorithm 7. Notice that in each iteration t, we call the solver

Algorithm 7 Accelerated Iterative Dual Random Projection (Acc-IDRP)—Primal Version.

```
Input: Data $\mathbf{X}, \mathbf{y}$, projection matrix $\mathbf{R}$.
Initialization: $\widehat{\mathbf{w}}_{\text{DRP}}^{(0)} = \mathbf{0}, \widehat{\boldsymbol{\alpha}}_{\text{DRP}}^{(0)} = \mathbf{0}, \mathbf{r}^{(0)} = -\mathbf{y}$.
Compute $\mathbf{z}^{(0)}$ by solving (4.18), and update $\mathbf{u}^{(0)} = \mathbf{r}^{(0)} - \mathbf{XRz}^{(0)}$, $\mathbf{p}^{(0)} = -\mathbf{u}^{(0)}$, $\mathbf{v}^{(0)} = \left(\frac{\mathbf{XX}^\top}{n} + \lambda\mathbf{I}_n\right)\mathbf{p}^{(0)}$.
for $t = 0, 1, 2, \ldots$ do
    Calculate $a^{(t)} = \frac{\langle \mathbf{r}^{(t)}, \mathbf{u}^{(t)} \rangle}{\langle \mathbf{p}^{(t)}, \mathbf{v}^{(t)} \rangle}$
    Update the dual approximation by $\widehat{\boldsymbol{\alpha}}_{\text{DRP}}^{(t+1)} = \widehat{\boldsymbol{\alpha}}_{\text{DRP}}^{(t)} + a^{(t)}\mathbf{p}^{(t)}$.
    Update primal approximation: $\widehat{\mathbf{w}}_{\text{DRP}}^{(t+1)} = \frac{1}{\lambda n}\mathbf{X}^\top \widehat{\boldsymbol{\alpha}}_{\text{DRP}}^{(t+1)}$.
    Update $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} + a^{(t)}\mathbf{v}^{(t)}$.
    Solve the projected problem in (4.18) and obtain solution $\widehat{\mathbf{z}}^{(t+1)}$.
    Update $\mathbf{u}^{(t+1)} = \mathbf{r}^{(t+1)} - \mathbf{XR}\widehat{\mathbf{z}}^{(t+1)}$.
    Update $\beta^{(t+1)} = \frac{\langle \mathbf{r}^{(t+1)}, \mathbf{u}^{(t)} \rangle}{\langle \mathbf{r}^{(t)}, \mathbf{r}^{(t)} \rangle}$.
    Update $\mathbf{p}^{(t+1)} = -\mathbf{u}^{(t+1)} + \beta^{(t+1)}\mathbf{p}^{(t)}$.
    Update $\mathbf{v}^{(t+1)} = \left(\frac{\mathbf{XX}^\top}{n} + \lambda\mathbf{I}_n\right)\mathbf{p}^{(t+1)}$.
end
```

for the following randomly projected problem based on the residual $\mathbf{r}^{(t)}$:

$$\widehat{\mathbf{z}}^{(t)} = \arg\min_{\mathbf{z}\in\mathbb{R}^d} \mathbf{z}^\top \left(\frac{\mathbf{R}^\top\mathbf{X}^\top\mathbf{XR}}{2n} + \frac{\lambda}{2}\mathbf{I}_d\right)\mathbf{z} - \langle \mathbf{R}^\top\mathbf{X}^\top\mathbf{r}^{(t)}, \mathbf{z}\rangle. \tag{4.18}$$

The accelerated IDRP algorithm runs the Acc-IHS Algorithm 5 in the dual space. However, Acc-IDRP is still a primal algorithm, since it updates the corresponding dual variables after solving the randomly projected primal problem (4.18). The dual version of Acc-IDRP algorithm would at each iteration solve the following dual optimization problem

$$\min_{\mathbf{u}\in\mathbb{R}^n} \mathbf{u}^\top \left(\frac{\mathbf{XRR}^\top\mathbf{X}^\top}{2n} + \frac{\lambda}{2}\mathbf{I}_n\right)\mathbf{u} - \langle \mathbf{r}^{(t)}, \mathbf{u}\rangle, \tag{4.19}$$

where $\mathbf{r}^{(t)}$ is the dual residual. This, however, is not a practical algorithm as it requires solving relatively more expensive dual problem. We present it in Algorithm 8 as it is easier to understand since it directly borrows the ideas of Acc-IHS described in Section 4.2.2.

Algorithm 8 Accelerated Iterative Dual Random Projection (Acc-IDRP)—Dual Version.

```
Input: Data $\mathbf{X}, \mathbf{y}$, projection matrix $\mathbf{R}$.
Initialization: $\widehat{\mathbf{w}}^{(0)}_{\text{DRP}} = \mathbf{0}, \widehat{\boldsymbol{\alpha}}^{(0)}_{\text{DRP}} = \mathbf{0}, \mathbf{r}^{(0)} = -\mathbf{y}$.
Compute $\mathbf{u}^{(0)}$ by solving (4.19), and update $\mathbf{p}^{(0)} = -\mathbf{u}^{(0)}$, $\mathbf{v}^{(0)} = \left(\frac{\mathbf{X}\mathbf{X}^\top}{n} + \lambda \mathbf{I}_n\right) \mathbf{p}^{(0)}$.
for $t = 0, 1, 2, \dots$ do
    Calculate $a^{(t)} = \frac{\langle \mathbf{r}^{(t)}, \mathbf{u}^{(t)} \rangle}{\langle \mathbf{p}^{(t)}, \mathbf{v}^{(t)} \rangle}$
    Update the dual approximation by $\widehat{\boldsymbol{\alpha}}^{(t+1)}_{\text{DRP}} = \widehat{\boldsymbol{\alpha}}^{(t)}_{\text{DRP}} + a^{(t)} \mathbf{p}^{(t)}$.
    Update primal approximation: $\widehat{\mathbf{w}}^{(t+1)}_{\text{DRP}} = \frac{1}{\lambda n} \mathbf{X}^\top \widehat{\boldsymbol{\alpha}}^{(t+1)}_{\text{DRP}}$.
    Update $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} + a^{(t)} \mathbf{v}^{(t)}$.
    Update $\mathbf{u}^{(t+1)}$ by solving (4.19).
    Update $\beta^{(t+1)} = \frac{\langle \mathbf{r}^{(t+1)}, \mathbf{u}^{(t)} \rangle}{\langle \mathbf{r}^{(t)}, \mathbf{r}^{(t)} \rangle}$.
    Update $\mathbf{p}^{(t+1)} = -\mathbf{u}^{(t+1)} + \beta^{(t+1)} \mathbf{p}^{(t)}$.
    Update $\mathbf{v}^{(t+1)} = \left(\frac{\mathbf{X}\mathbf{X}^\top}{n} + \lambda \mathbf{I}_n\right) \mathbf{p}^{(t+1)}$.
end
```

Though the computational cost per iteration of Acc-IDRP and standard IDRP are the same, Acc-IDRP has two main advantages over IDRP. First, as a preconditioned conjugate gradient procedure, Acc-IDRP is guaranteed to converge, and reach the optimum $\mathbf{w}^*$ within n iterations, even when the projection dimension d is very small. Second, even when the projection dimension d is large enough for the standard IDRP converge quickly to the optimum, Acc-IDRP converges even faster.

4.2.6 Primal-Dual sketch for big and high-dimensional problems

Approach	Suitable Situation	Reduced Dimension	Recovery	Iterative
`Classical Sketch`	large n, small p	sample reduction	×	×
`Random Projection`	small n, large p	dimension reduction	×	×
`Hessian Sketch`	large n, small p	sample reduction	✓	✓
`DRP`	small n, large p	dimension reduction	✓	✓

Table 4.1: Comparison of various algorithms for data sketching in solving large-scale problems.

In this section, we combine the idea of the iterative Hessian sketch and iterative dual random projection from the primal-dual point of view. We propose a more efficient sketching technique named *Iterative Primal-Dual Sketch* (IPDS), which simultaneously reduces the sample size and dimensionality of the problem, while recovering the original solution to a high precision.

The Hessian sketch is particularly suitable for the case where the sample size is much larger than the problem dimension and the computational bottleneck is "big n". Here the Hessian sketch reduces the sample size significantly, and as a consequence, speeds up the computation. By utilizing the iterative extension approximation error can be further reduced to recover the original solution to a high precision. In contrast, the dual random projection is aimed at dimensionality reduction and is suitable for the case of high-dimensional data, with relatively small sample size. Here the computational bottleneck is "large p" and the random projection is used to reduce dimensionality and speedup computations.

Hessian sketch is particularly suitable for the case where sample size is much larger than problem dimension, where the computational bottleneck is "big n". Therefore, it is possible to use Hessian sketch to reduce the sample size significantly and consequently speed up the computation. It also possible to utilize the iterative extension to reduce the approximation error further to recover the original solution to a high precision. In contrast, dual random projection is aimed at dimension reduction and is mostly suitable for the case of high-dimensional data but relatively small sample size, where the computational bottleneck is "large p", and we would like to use random projection to perform dimension reduction and gain speedup. Table 7.1 summarizes these characteristics.

As shown in Table 7.1, the Hessian sketch and dual random projection are suitable for problems where the number of observations n and the number of variables p are not balanced, that is, when one is much larger than the other. Modern massive data-sets have a characteristic that both n and p are very large, for example, the click-through rate (CTR)

Algorithm 9 Iterative Primal-Dual Sketch (IPDS).

```
Input: Data $\mathbf{X} \in \mathbb{R}^{n\times p}, \mathbf{y} \in \mathbb{R}^n$, sketching matrix $\mathbf{R} \in \mathbb{R}^{p\times d}, \mathbf{\Pi} \in \mathbb{R}^{n\times m}$.
Initialization: $\widehat{\mathbf{w}}_{\text{DS}}^{(0)} = \mathbf{0}$.
for $t = 0, 1, 2, \ldots$ do
    Initialization: $\widetilde{\mathbf{z}}^{(0)} = \mathbf{0}, k = 0$
    if not converged then
        Solve the sketched problem in (4.21) and obtain solution $\Delta\mathbf{z}^{(k)}$.
        Update $\widetilde{\mathbf{z}}^{(k+1)} = \widetilde{\mathbf{z}}^{(k)} + \Delta\mathbf{z}^{(k)}$.
        Update $k = k + 1$.
    end
    Update dual approximation: $\widehat{\boldsymbol{\alpha}}_{\text{DS}}^{(t+1)} = \mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}_{\text{DS}}^{(t)} - \mathbf{X}\mathbf{R}\widetilde{\mathbf{z}}^{(k+1)}$.
    Update primal approximation: $\widehat{\mathbf{w}}_{\text{DS}}^{(t+1)} = \frac{1}{\lambda n}\mathbf{X}^\top \widehat{\boldsymbol{\alpha}}_{\text{DS}}^{(t+1)}$.
end
```

prediction data sets provided by Criteo[1] has $n \geq 4\times 10^9$ and $p \geq 8\times 10^8$. To tackle problems of this size, it is desirable to have a sketching method capable of simultaneously reducing the sample size and dimensionality.

Inspired by the primal-dual view described in Section 4.2.4, we propose the iterative Primal-Dual Sketch. The iterative Primal-Dual Sketch only involves solving small scale problems where the sample size and dimension are both small. For the original problem (4.1) with data $\{\mathbf{X}, \mathbf{y}\}$, we first construct the randomly projected data, as well as the *doubly sketched* data, as follows:

$$\mathbf{X} \to \mathbf{X}\mathbf{R}, \qquad \mathbf{X}\mathbf{R} \to \mathbf{\Pi}^\top \mathbf{X}\mathbf{R},$$

where $\mathbf{X}\mathbf{R}$ is the randomly projected data, and $\mathbf{\Pi}^\top\mathbf{X}\mathbf{R}$ is doubly sketched data. Let $\widehat{\mathbf{w}}_{\text{DS}}^{(0)} = \mathbf{0}$. At every iteration of IPDS, we first apply random projection on the primal problem (which is equivalent to the Hessian sketch on the dual problem), and obtain the following

1. available at `http://labs.criteo.com/downloads/download-terabyte-click-logs/`

problem:

$$\min_{\mathbf{z}\in\mathbb{R}^d} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}_{\text{DS}}^{(t)} - \mathbf{X}\mathbf{R}\mathbf{z}||_2^2 + \frac{\lambda}{2}||\mathbf{z} + \mathbf{R}^\top\widehat{\mathbf{w}}_{\text{DS}}^{(t)}||_2^2, \tag{4.20}$$

which is the same as the iterative dual random projection subproblem (4.15). However, different from IDRP, we do not directly solve (4.20). Instead, we apply the iterative Hessian sketch to find an approximate solution to

$$\min_{\mathbf{z}\in\mathbb{R}^d} \mathbf{z}^\top \left(\frac{\mathbf{R}^\top\mathbf{X}^\top\mathbf{X}\mathbf{R}}{2n} + \frac{\lambda}{2}\mathbf{I}_d\right)\mathbf{z} - \left\langle \frac{(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}_{\text{DS}}^{(t)})^\top\mathbf{X}\mathbf{R}}{n} - \lambda\mathbf{R}^\top\widehat{\mathbf{w}}_{\text{DS}}^{(t)}, \mathbf{z}\right\rangle.$$

Let $\widetilde{\mathbf{z}}^{(0)} = \mathbf{0}$. At iteration k in the inner loop, we solve the following sketched problem:

$$\begin{aligned}\Delta\mathbf{z}^{(k)} =& \arg\min_{\Delta\mathbf{z}} \Delta\mathbf{z}^\top \left(\frac{\mathbf{R}^\top\mathbf{X}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{X}\mathbf{R}}{2n} + \frac{\lambda}{2}\mathbf{I}_d\right)\mathbf{z} \\ &- \left\langle \frac{\mathbf{R}^\top\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}_{\text{DS}}^{(t)} - \mathbf{X}\mathbf{R}\widetilde{\mathbf{z}}^{(k)})}{n} - \lambda\mathbf{R}^\top\widehat{\mathbf{w}}_{\text{DS}}^{(t)} - \lambda\widetilde{\mathbf{z}}^{(k)}, \Delta\mathbf{z}\right\rangle \end{aligned} \tag{4.21}$$

and update $\widetilde{\mathbf{z}}^{(k+1)}$ as

$$\widetilde{\mathbf{z}}^{(k+1)} = \widetilde{\mathbf{z}}^{(k)} + \Delta\mathbf{z}^{(k)}.$$

The key point is that for the subproblem (4.21), the sketched data matrix is only of size $m \times d$, compared to the original problem size $n \times p$, where $n \gg m, p \gg d$. In contrast, the IHS still needs to solve sub-problems of size $m \times p$, while IDRP needs to solve sub-problems of size $n \times d$. We only need to call solvers of $m \times d$ problem (4.21) logarithmic times to obtain a solution of high approximation quality.

The pseudo code of Iterative Primal-Dual Sketch (IPDS) is summarized in Algorithm 9. It is also possible to perform iterative Primal-Dual Sketch via another direction, that is, first perform primal Hessian sketch, and then apply dual Hessian sketch to solve the sketched primal problem:

$$\mathbf{X} \rightarrow \mathbf{\Pi}^\top \mathbf{X}, \qquad \mathbf{\Pi}^\top \mathbf{X} \rightarrow \mathbf{\Pi}^\top \mathbf{X}\mathbf{R}.$$

The idea presented in Section 4.2.2 can also be adopted to further reduce the number of calls to $m \times d$ scale sub-problems, which leads to the accelerated iterative primal-dual sketch (Acc-IPDS) algorithm, of which the pseudo code is summarized in Algorithm 10. In Acc-IPDS, we maintain both the vectors in the primal space $\mathbf{u}_\mathrm{P}, \mathbf{v}_\mathrm{P}, \mathbf{r}_\mathrm{P}$ and the vectors in the dual space $\mathbf{u}_\mathrm{D}, \mathbf{v}_\mathrm{D}, \mathbf{r}_\mathrm{D}$, to make sure that the updating directions for both primal variables and dual variables are conjugate with previous updating directions. Moreover, based on the residual vector $\mathbf{r}_\mathrm{P}$, Acc-IPDS iteratively calls the solver to find a solution of the following sketched linear system of scale $m \times d$:

$$\widehat{\mathbf{u}}_P^{(k)} = \arg\min_{\mathbf{u}} \mathbf{u}^\top \left(\frac{\mathbf{R}^\top \mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}\mathbf{R}}{2n} + \frac{\lambda}{2}\mathbf{I}_d \right) \mathbf{u} - \left\langle \mathbf{r}_\mathrm{P}^{(k)}, \mathbf{u} \right\rangle . \tag{4.22}$$

As we will show in the subsequent section, the number of calls for solving problem (4.22) only grows logarithmically with the inverse of approximation error.

4.3 Main theoretical results

We present theoretical properties of various iterative sketching procedures, while the proofs are deferred to Section 5.5. First, we provide a unified analysis of the Hessian sketch and dual random projection. The unified analysis basically follows the analysis of [Zhang et al., 2014] and [Pilanci and Wainwright, 2016], but provides recovery guarantees for both the *primal* and *dual* variables of interest, simultaneously. Next, we present convergence analysis for the proposed accelerated IHS and IDRP algorithms. We show improved convergence speed compared to the standard IHS and IDRP algorithms. Finally, we prove that the iterative primal-dual sketch converges to the optimum with the number of iterations growing only logarithmically with the target approximation accuracy. This makes the proposed

Algorithm 10 Accelerated Iterative Primal-Dual Sketch (Acc-IPDS).

```
**Input:** Data $\mathbf{X} \in \mathbb{R}^{n\times p}, \mathbf{y} \in \mathbb{R}^n$, sketching matrix $\mathbf{R} \in \mathbb{R}^{p\times d}, \mathbf{\Pi} \in \mathbb{R}^{n\times m}$.
**Initialization:** $\widehat{\mathbf{w}}_{\text{DS}}^{(0)} = \mathbf{0}, \widehat{\boldsymbol{\alpha}}_{\text{DS}}^{(0)} = \mathbf{0}, \mathbf{r}_{\text{Dual}}^{(0)} = -\mathbf{y}$.
**Initialization:** $k = 0, \widetilde{\mathbf{z}}^{(k)} = \mathbf{0}, \mathbf{r}_{\text{P}}^{(0)} = \mathbf{R}^\top\mathbf{X}^\top\mathbf{r}_{\text{D}}^{(0)}$.
Compute $\widehat{\mathbf{u}}_{\text{P}}^{(0)}$ by solving (4.22), and update $\mathbf{p}_{\text{P}}^{(0)} = -\mathbf{u}_{\text{P}}^{(0)}$, calculate $\mathbf{v}_{\text{P}}^{(0)} = \left(\frac{\mathbf{R}^\top\mathbf{X}\mathbf{X}\mathbf{R}}{n} + \lambda\mathbf{I}_d\right)\mathbf{p}_{\text{P}}^{(0)}$.
**if** *not converged* **then**
    Calculate $a_{\text{P}}^{(k)} = \frac{\langle \mathbf{r}_{\text{P}}^{(k)}, \mathbf{u}_{\text{P}}^{(k)}\rangle}{\langle \mathbf{p}_{\text{P}}^{(k)}, \mathbf{v}_{\text{P}}^{(k)}\rangle}$, and update the approximation by $\widetilde{\mathbf{z}}^{(k+1)} = \widetilde{\mathbf{z}}^{(k)} + a_{\text{P}}^{(k)}\mathbf{p}_{\text{P}}^{(k)}$.
    Update $\mathbf{r}_{\text{P}}^{(k+1)} = \mathbf{r}_{\text{P}}^{(k)} + a_{\text{P}}^{(k)}\mathbf{v}^{(k)}$, and update $\mathbf{u}_{\text{P}}^{(k+1)}$ by solving (4.22).
    Update $\beta_{\text{P}}^{(k+1)} = \frac{\langle \mathbf{r}_{\text{P}}^{(k+1)}, \mathbf{u}_{\text{P}}^{(k)}\rangle}{\langle \mathbf{r}_{\text{P}}^{(k)}, \mathbf{r}_{\text{P}}^{(k)}\rangle}$, and update $\mathbf{p}_{\text{P}}^{(k+1)} = -\mathbf{u}_{\text{P}}^{(k+1)} + \beta_{\text{P}}^{(k+1)}\mathbf{p}_{\text{P}}^{(k)}$.
    Update $\mathbf{v}_{\text{P}}^{(k+1)} = \left(\frac{\mathbf{R}^\top\mathbf{X}^\top\mathbf{X}\mathbf{R}}{n} + \lambda\mathbf{I}_p\right)\mathbf{p}_{\text{P}}^{(t+1)}$, and update $k = k+1$.
**end**
Compute $\mathbf{u}_{\text{D}}^{(0)} = \mathbf{r}_{\text{D}}^{(0)} - \mathbf{X}\mathbf{R}\widetilde{\mathbf{z}}^{(k+1)}$, $\mathbf{p}_{\text{D}}^{(0)} = -\mathbf{u}_{\text{D}}^{(0)}$, $\mathbf{v}_{\text{D}}^{(0)} = \left(\frac{\mathbf{X}\mathbf{X}^\top}{n} + \lambda\mathbf{I}_n\right)\mathbf{p}_{\text{D}}^{(0)}$.
**for** $t = 0, 1, 2, \ldots$ **do**
    Calculate $a_{\text{D}}^{(t)} = \frac{\langle \mathbf{r}_{\text{D}}^{(t)}, \mathbf{u}_{\text{D}}^{(t)}\rangle}{\langle \mathbf{p}_{\text{D}}^{(t)}, \mathbf{v}_{\text{D}}^{(t)}\rangle}$, and update the dual approximation by $\widehat{\boldsymbol{\alpha}}_{\text{DS}}^{(t+1)} = \widehat{\boldsymbol{\alpha}}_{\text{DS}}^{(t)} + a_{\text{D}}^{(t)}\mathbf{p}_{\text{D}}^{(t)}$.
    Update primal approximation: $\widehat{\mathbf{w}}_{\text{DS}}^{(t+1)} = \frac{1}{\lambda n}\mathbf{X}^\top\widehat{\boldsymbol{\alpha}}_{\text{DS}}^{(t+1)}$, and update $\mathbf{r}_{\text{D}}^{(t+1)} = \mathbf{r}_{\text{D}}^{(t)} + a_{\text{D}}^{(t)}\mathbf{v}_{\text{D}}^{(t)}$.
    **Initialization:** $k = 0, \widetilde{\mathbf{z}}^{(k)} = \mathbf{0}, \mathbf{r}_{\text{P}}^{(0)} = \mathbf{R}^\top\mathbf{X}^\top\mathbf{r}_{\text{D}}^{(t+1)}$.
    Compute $\widehat{\mathbf{u}}_{\text{P}}^{(0)}$ by solving (4.22), and update $\mathbf{p}_{\text{P}}^{(0)} = -\mathbf{u}_{\text{P}}^{(0)}$, calculate $\mathbf{v}_{\text{P}}^{(0)} = \left(\frac{\mathbf{R}^\top\mathbf{X}\mathbf{X}\mathbf{R}}{n} + \lambda\mathbf{I}_d\right)\mathbf{p}_{\text{P}}^{(0)}$.
    **if** *not converged* **then**
        Calculate $a_{\text{P}}^{(k)} = \frac{\langle \mathbf{r}_{\text{P}}^{(k)}, \mathbf{u}_{\text{P}}^{(k)}\rangle}{\langle \mathbf{p}_{\text{P}}^{(k)}, \mathbf{v}_{\text{P}}^{(k)}\rangle}$, and update the approximation by $\widetilde{\mathbf{z}}^{(k+1)} = \widetilde{\mathbf{z}}^{(k)} + a_{\text{P}}^{(k)}\mathbf{p}_{\text{P}}^{(k)}$.
        Update $\mathbf{r}_{\text{P}}^{(k+1)} = \mathbf{r}_{\text{P}}^{(k)} + a_{\text{P}}^{(k)}\mathbf{v}^{(k)}$, and update $\mathbf{u}_{\text{P}}^{(k+1)}$ by solving (4.22).
        Update $\beta_{\text{P}}^{(k+1)} = \frac{\langle \mathbf{r}_{\text{P}}^{(k+1)}, \mathbf{u}_{\text{P}}^{(k)}\rangle}{\langle \mathbf{r}_{\text{P}}^{(k)}, \mathbf{r}_{\text{P}}^{(k)}\rangle}$, and update $\mathbf{p}_{\text{P}}^{(k+1)} = -\mathbf{u}_{\text{P}}^{(k+1)} + \beta_{\text{P}}^{(k+1)}\mathbf{p}_{\text{P}}^{(k)}$.
        Update $\mathbf{v}_{\text{P}}^{(k+1)} = \left(\frac{\mathbf{R}^\top\mathbf{X}^\top\mathbf{X}\mathbf{R}}{n} + \lambda\mathbf{I}_p\right)\mathbf{p}_{\text{P}}^{(t+1)}$, and update $k = k+1$.
    **end**
    Update $\mathbf{u}_{\text{D}}^{(t+1)} = \mathbf{r}_{\text{D}}^{(t+1)} - \mathbf{X}\mathbf{R}\widetilde{\mathbf{z}}_{\text{D}}^{(k+1)}$, and update $\beta_{\text{D}}^{(t+1)} = \frac{\langle \mathbf{r}_{\text{D}}^{(t+1)}, \mathbf{u}_{\text{D}}^{(t)}\rangle}{\langle \mathbf{r}_{\text{D}}^{(t)}, \mathbf{r}_{\text{D}}^{(t)}\rangle}$.
    Update $\mathbf{p}_{\text{D}}^{(t+1)} = -\mathbf{u}_{\text{D}}^{(t+1)} + \beta_{\text{D}}^{(t+1)}\mathbf{p}_{\text{D}}^{(t)}$, and update $\mathbf{v}_{\text{D}}^{(t+1)} = \left(\frac{\mathbf{X}\mathbf{X}^\top}{n} + \lambda\mathbf{I}_n\right)\mathbf{p}_{\text{D}}^{(t+1)}$.
**end**
```

primal-dual sketching schema suitable for large-scale learning problems with huge number of features.

4.3.1 A unified analysis of Hessian sketch and dual random projection

In this section we provide a simple and unified analysis for the recovery performance of the Hessian sketch and dual random projection. First, we define the following notion of the *Gaussian width*, which will be useful in the statement of our results. For any set $\mathcal{K} \subseteq \mathbb{R}^p$, the Gaussian width is defined as:

$$\mathbb{W}(\mathcal{K}) = \mathbb{E}\left[\sup_{\mathbf{w}\in\mathcal{K}} \langle \mathbf{w}, \mathbf{g} \rangle\right], \tag{4.23}$$

where $\mathbf{g}$ is a random vector drawn from a Normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_p)$. Intuitively, if the set $\mathcal{K}$ is restricted to certain directions and magnitude, then $\mathbb{W}(\mathcal{K})$ will be small [Vershynin, 2015]. Given a set $\mathcal{K}$ and a random matrix $\mathbf{R} \in \mathbb{R}^{p\times d}$, and a unit-length vector $\mathbf{v} \in \mathbb{R}^p$, the following quantities will be important:

$$\begin{aligned}
\rho_1(\mathcal{K}, \mathbf{R}) &= \inf_{\mathbf{u}\in\mathcal{K}\cap\mathcal{S}^{p-1}} \mathbf{u}^\top \mathbf{R}\mathbf{R}^\top \mathbf{u} \\
\rho_2(\mathcal{K}, \mathbf{R}, \mathbf{v}) &= \sup_{\mathbf{u}\in\mathcal{K}\cap\mathcal{S}^{p-1}} \left|\mathbf{u}^\top \left(\mathbf{R}\mathbf{R}^\top - \mathbf{I}_p\right)\mathbf{v}\right|,
\end{aligned}$$

where $\mathcal{S}^{p-1} = \{\mathbf{x} \in \mathbb{R}^p \mid \|\mathbf{x}\| = 1\}$ is the p-dimensional Euclidean unit-sphere. The sketching matrix $\mathbf{R}$ will be constructed to satisfy

$$\mathbb{E}\left[\mathbf{R}\mathbf{R}^\top\right] = \mathbf{I}_p,$$

and, as the sketching dimension d increases, the matrix $\mathbf{R}\mathbf{R}^\top$ will get closer to $\mathbf{I}_p$. Thus, we would like to have $\rho_1(\mathcal{K}, \mathbf{R})$ be close to 1, and $\rho_2(\mathcal{K}, \mathbf{R}, \mathbf{v})$ be close to 0. To simplify presentation, assume that the entries of a random matrix $\mathbf{R}$ are sampled i.i.d. from a sub-Gaussian distribution and $\mathbb{E}\left[\mathbf{R}\mathbf{R}^\top\right] = \mathbf{I}_p$. The following lemma states how large the sketching dimension d should be in order to control $\rho_1(\mathcal{K}, \mathbf{R})$ and $\rho_2(\mathcal{K}, \mathbf{R}, \mathbf{v})$.

Lemma 23 ([Pilanci and Wainwright, 2015])**.** *When* $\mathbf{R}$ *is sampled i.i.d. from a sub-Gaussian distribution and* $\mathbb{E}\left[\mathbf{R}\mathbf{R}^\top\right] = \mathbf{I}_p$*, then there exists universal constants* C_0 *such that*

$$\rho_1(\mathcal{K}, \mathbf{R}) \geq 1 - C_0 \sqrt{\frac{\mathbb{W}^2(\mathcal{K} \cap \mathcal{S}^{p-1})}{d}} \log\left(\frac{1}{\delta}\right) \tag{4.24}$$

and

$$\rho_2(\mathcal{K}, \mathbf{R}, \mathbf{v}) \leq C_0 \sqrt{\frac{\mathbb{W}^2(\mathcal{K} \cap \mathcal{S}^{p-1})}{d}} \log\left(\frac{1}{\delta}\right), \tag{4.25}$$

with probability at least $1 - \delta$*.*

For a set $\mathcal{K} \subseteq \mathbb{R}^p$, define the transformed set $\mathbf{X}\mathcal{K}$ with $\mathbf{X} \in \mathbb{R}^{n \times p}$ as

$$\mathbf{X}\mathcal{K} = \{\mathbf{u} = \mathbf{X}\mathbf{v} \in \mathbb{R}^n \mid \mathbf{v} \in \mathcal{K}\}.$$

Before presenting the main unifying result, let us recall the reductions in the Hessian sketch and dual random projection. For the Hessian sketch, we perform *sample reduction* with the transformation $\mathbf{X} \to \mathbf{\Pi}^\top\mathbf{X}$, while for the dual random projection, we perform *dimension reduction* with the transformation $\mathbf{X} \to \mathbf{X}\mathbf{R}$, where $\mathbf{\Pi} \in \mathbb{R}^{n \times m}$ and $\mathbf{R} \in \mathbb{R}^{p \times d}$. Let $\widehat{\mathbf{w}}_{\text{HS}}$ be an approximate solution obtained via the Hessian sketch by solving (4.4). The corresponding dual variables are obtained using the following transformation

$$\widehat{\boldsymbol{\alpha}}_{\text{HS}} = \mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}_{\text{HS}}.$$

Likewise, let $\widehat{\boldsymbol{\alpha}}_{\text{DRP}}$ and $\widehat{\mathbf{w}}_{\text{DRP}}$ be the approximate dual and primal variables obtained by the dual random projection. The following theorem established the recovery bound for

$\widehat{\boldsymbol{\alpha}}_{\text{HS}}, \widehat{\boldsymbol{\alpha}}_{\text{DRP}}$ and $\widehat{\mathbf{w}}_{\text{HS}}, \widehat{\mathbf{w}}_{\text{DRP}}$ simultaneously.

Theorem 24. *Suppose we perform the Hessian sketch or the dual random projection for the problem (4.1) with a sub-Gaussian sketching matrix $\mathbf{\Pi} \in \mathbb{R}^{n\times m}$ (for HS) or $\mathbf{R} \in \mathbb{R}^{p\times d}$ (for DRP), satisfying $\mathbb{E}\left[\mathbf{R}\mathbf{R}^\top\right] = \mathbf{I}_p$ and $\mathbb{E}\left[\mathbf{\Pi}\mathbf{\Pi}^\top\right] = \mathbf{I}_n$. Then there exists a universal constant C_0 such that with probability at least $1-\delta$, the following approximation error bounds hold for HS or DRP:* ***For Hessian sketch:***

$$||\widehat{\mathbf{w}}_{\text{HS}} - \mathbf{w}^*||_{\mathbf{X}} \leq \frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}}\log\left(\frac{1}{\delta}\right)}{1-C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}}\log\left(\frac{1}{\delta}\right)}||\mathbf{w}^*||_{\mathbf{X}}, \quad \textit{and} \tag{4.26}$$

$$||\widehat{\boldsymbol{\alpha}}_{\text{HS}} - \boldsymbol{\alpha}^*||_2 \leq \frac{\sqrt{n}C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}}\log\left(\frac{1}{\delta}\right)}{1-C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}}\log\left(\frac{1}{\delta}\right)}||\mathbf{w}^*||_{\mathbf{X}}. \tag{4.27}$$

For dual random projection:

$$||\widehat{\mathbf{w}}_{\text{DRP}} - \mathbf{w}^*||_2 \leq \frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}{1-C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}||\mathbf{w}^*||_2, \quad \textit{and} \tag{4.28}$$

$$||\widehat{\boldsymbol{\alpha}}_{\text{DRP}} - \boldsymbol{\alpha}^*||_{\mathbf{X}^\top} \leq \frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}{1-C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}||\boldsymbol{\alpha}^*||_{\mathbf{X}^\top}. \tag{4.29}$$

Theorem 24 is proven in Appendix 4.6.1. We have the following remarks on Theorem 24.

Remark 1. *For a general low-dimensional problem where $n \gg p$, $\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1}) = p$. Thus we have $||\widehat{\mathbf{w}}_{\text{HS}} - \mathbf{w}^*||_{\mathbf{X}} \lesssim \sqrt{\frac{p}{m}}\log\left(\frac{1}{\delta}\right)||\mathbf{w}^*||_{\mathbf{X}}$. This is the recovery bound proved in [Proposition 1, Pilanci and Wainwright, 2016].*

Remark 2. *For high-dimensional problems when p is large, $\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1}) = n$. Thus we have $||\widehat{\mathbf{w}}_{\text{DRP}} - \mathbf{w}^*||_2 \lesssim \sqrt{\frac{n}{d}}\log\left(\frac{1}{\delta}\right)||\mathbf{w}^*||_2$. Moreover, when $\mathbf{X}$ is low-rank, that is* rank$(\mathbf{X}) = r$ *and* $r \ll \min(n,p)$*, we have* $\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1}) = r$*, leading to* $||\widehat{\mathbf{w}}_{\mathrm{DRP}} - \mathbf{w}^*||_2 \lesssim \sqrt{\frac{r}{d}}\log\left(\frac{1}{\delta}\right)||\mathbf{w}^*||_2$*. This recovery bound removes a* $\sqrt{\log r}$ *factor from a bound obtained in Theorem 1 of [Zhang et al., 2014].*

Analysis of IHS and DRP when $\mathbf{X}$ is approximately low-rank

In this section we provide recovery guarantees for the case when the data matrix $\mathbf{X}$ is *approximately* low rank. To make $\mathbf{X}$ be well approximated by a rank r matrix where $r \ll \min(n,p)$, we assume σ_{r+1}, the $r+1$-th singular value of $\mathbf{X}$, is small enough. Let us decompose $\mathbf{X}$ into two parts as

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^\top + \mathbf{U}_{\bar{r}}\mathbf{\Sigma}_{\bar{r}}\mathbf{V}_{\bar{r}}^\top = \mathbf{X}_r + \mathbf{X}_{\bar{r}},$$

where $\mathbf{X}_r$ corresponds to the largest r singular values and $\mathbf{X}_{\bar{r}}$ corresponds to the remaining singular values with $\bar{r} = \{r+1, ..., \max(n,p)\}$. Furthermore, suppose that the solution to (4.1), $\mathbf{w}^*$, is well approximated by the a linear combination of the top r left singular vectors of $\mathbf{X}$ and that the remaining singular vectors are almost orthogonal to $\mathbf{w}^*$. We can represent this notion more formally depending on the particular method. For the Hessian sketch we assume that for some ρ we have

$$||\mathbf{X}_{\bar{r}}\mathbf{w}^*||_2 \leq \rho||\mathbf{X}\mathbf{w}^*||_2,$$

while for the dual random projection we assume that

$$||\mathbf{V}_{\bar{r}}^\top\mathbf{w}^*||_2 \leq \varrho||\mathbf{w}^*||_2,$$

for some ϱ. For simplicity, assume that the entries of the sketching matrix $\mathbf{\Pi} \in \mathbb{R}^{m\times n}$ and $\mathbf{R} \in \mathbb{R}^{p\times d}$ are sampled i.i.d. from a zero-mean sub-Gaussian distributions, and satisfying

$\mathbb{E}\left[\mathbf{R}\mathbf{R}^\top\right] = \mathbf{I}_p$ and $\mathbb{E}\left[\mathbf{\Pi}\mathbf{\Pi}^\top\right] = \mathbf{I}_n$ respectively. We have the following recovery bounds for the Hessian sketch and the dual random projection.

Theorem 25. *With probability at least $1-\delta$, we have:*

For the Hessian sketch:

$$m \geq \max\left(32(r+1), 4\log\left(\frac{2m}{\delta}\right), \frac{784\sigma_{r+1}^2}{9\lambda}\right)\log\left(\frac{n}{\delta}\right)$$

then

$$||\widehat{\mathbf{w}}_{\text{HS}} - \mathbf{w}^*||_{\mathbf{X}} \leq 4\sqrt{\frac{1}{1-\epsilon_1} + \frac{\sigma_{r+1}^2}{\lambda n}} \cdot \sqrt{\frac{\epsilon_1^2 + \tau_1^2\rho^2}{1-\epsilon_1} + \frac{\tau_1^2\sigma_{r+1}^2 + \rho^2\upsilon_1^2\sigma_{r+1}^2}{\lambda n}}||\mathbf{w}^*||_{\mathbf{X}};$$

For the dual random projection: *if*

$$d \geq \max\left(32(r+1), 4\log\left(\frac{2d}{\delta}\right), \frac{784p\sigma_{r+1}^2}{9\lambda n}\right)\log\left(\frac{p}{\delta}\right)$$

then

$$||\widehat{\mathbf{w}}_{\text{DRP}} - \mathbf{w}^*||_2 \leq 4\sqrt{\frac{1}{1-\epsilon_2} + \frac{\sigma_{r+1}^2}{\lambda n}} \cdot \sqrt{\frac{\epsilon_2^2 + \tau_2^2\varrho^2}{1-\epsilon_2} + \frac{\tau_2^2\sigma_{r+1}^2 + \varrho^2\upsilon_2^2\sigma_{r+1}^2}{\lambda n}}||\mathbf{w}^*||_2, \quad (4.30)$$

where $\epsilon_1, \epsilon_2, \tau_1, \tau_2, \upsilon_1, \upsilon_2$ are defined as

$$\begin{aligned}
\epsilon_1 &= 2\sqrt{\frac{2(r+1)}{m}\log\frac{2r}{\delta}}, & \epsilon_2 &= 2\sqrt{\frac{2(r+1)}{d}\log\frac{2r}{\delta}},\\
\tau_1 &= \frac{7}{3}\sqrt{\frac{2(n-r)}{m}\log\frac{n}{\delta}}, & \tau_2 &= \frac{7}{3}\sqrt{\frac{2(p-r)}{d}\log\frac{p}{\delta}},\\
\upsilon_1 &= 2\sqrt{\frac{2(n-r+1)}{m}\log\frac{2(n-r)}{\delta}}, & \upsilon_2 &= 2\sqrt{\frac{2(p-r+1)}{d}\log\frac{2(p-r)}{\delta}}.
\end{aligned}$$

The proof is provided in Appendix 4.6.2. We make the following comments on Theorem

25.

Remark 3. *When $\sigma_{r+1} = 0$ and $\mathbf{X}$ is of exact rank r, the above result simplifies to*

$$||\widehat{\mathbf{w}}_{\mathrm{HS}} - \mathbf{w}^*||_{\mathbf{X}} \lesssim \sqrt{\frac{r}{m}}||\mathbf{w}^*||_{\mathbf{X}}, \qquad ||\widehat{\mathbf{w}}_{\mathrm{DRP}} - \mathbf{w}^*||_2 \lesssim \sqrt{\frac{r}{d}}||\mathbf{w}^*||_2 \tag{4.31}$$

These are the results of Theorem 24.

Remark 4. *We see that if we have σ_{r+1}, ρ, and ϱ sufficiently small, then the guarantees* (4.31) *still hold. In particular, for the Hessian sketch we need that*

$$\sigma_{r+1} \lesssim \sqrt{\lambda}, \qquad \rho \lesssim \sqrt{\frac{r}{n}},$$

while for the dual random projection we need

$$\sigma_{r+1} \lesssim \sqrt{\frac{\lambda n}{p}}, \qquad \varrho \lesssim \sqrt{\frac{r}{p}}.$$

4.3.2 Analysis of the accelerated IHS and IDRP methods

In this section we provide convergence analysis for the Acc-IHS and Acc-IDRP algorithms. Recall that Acc-IHS and Acc-IDRP algorithms are preconditioned conjugate gradient methods on primal and dual problems, with a sketched Hessian as a pre-conditioner. Therefore, we will use a classical analysis of preconditioned conjugate gradient [Luenberger, 1973] to obtain the following convergence guarantees.

Proposition 26. *Iterates obtain by Acc-IHS satisfy*

$$||\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)} - \mathbf{w}^*||_{\mathbf{X}} \leq 2\left(\frac{\sqrt{\kappa_{\mathrm{HS}}(\mathbf{X}, \mathbf{\Pi}, \lambda)} - 1}{\sqrt{\kappa_{\mathrm{HS}}(\mathbf{X}, \mathbf{\Pi}, \lambda)} + 1}\right)^t ||\mathbf{w}^*||_{\mathbf{X}}, \tag{4.32}$$

where

$$\kappa_{\mathrm{HS}}(\mathbf{X},\mathbf{\Pi},\lambda) =\kappa\left(\left(\frac{\mathbf{X}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{X}}{n}+\lambda\mathbf{I}_p\right)^{-1}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}+\lambda\mathbf{I}_p\right)\right).$$

Iterates obtain by Acc-IDRP satisfy

$$||\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}^{(t)}-\boldsymbol{\alpha}^*||_{\mathbf{X}^\top}\leq 2\left(\frac{\sqrt{\kappa_{\mathrm{DRP}}(\mathbf{X},\mathbf{R},\lambda)}-1}{\sqrt{\kappa_{\mathrm{DRP}}(\mathbf{X},\mathbf{R},\lambda)}+1}\right)^t||\boldsymbol{\alpha}^*||_{\mathbf{X}^\top}, \tag{4.33}$$

where

$$\kappa_{\mathrm{DRP}}(\mathbf{X},\mathbf{R},\lambda) =\kappa\left(\left(\frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top\mathbf{X}^\top}{n}+\lambda\mathbf{I}_n\right)^{-1}\left(\frac{\mathbf{X}\mathbf{X}^\top}{n}+\lambda\mathbf{I}_n\right)\right).$$

From Proposition 26, we see that the convergence of Acc-IHS and Acc-IDRP heavily depend on the respective condition number, $\kappa_{\mathrm{HS}}(\mathbf{X},\mathbf{\Pi},\lambda)$ or $\kappa_{\mathrm{DRP}}(\mathbf{X},\mathbf{R},\lambda)$. Therefore, it is crucial to obtain a refined upper bound on these condition numbers. We make use of the following result in [Mendelson et al., 2007].

Lemma 27. *Suppose the elements of $\mathbf{\Pi}\in\mathbb{R}^{n\times m}$ are sampled i.i.d. from a zero-mean sub-Gaussian distribution satisfying $\mathbb{E}\left[\mathbf{\Pi}\mathbf{\Pi}^\top\right]=\mathbf{I}_n$, then there exists a universal constant C_0 such that, for any subset $\mathcal{K}\subseteq\mathbb{R}^n$, we have*

$$\sup_{\mathbf{u}\in\mathcal{K}\cap\mathcal{S}^{n-1}}\left|\mathbf{u}^\top\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\mathbf{u}\right|\leq C_0\sqrt{\frac{\mathbb{W}^2(\mathcal{K})}{m}}\log\left(\frac{1}{\delta}\right)$$

with probability at least $1-\delta$.

An application of this lemma gives us the following bounds on the condition numbers $\kappa_{\mathrm{HS}}(\mathbf{X},\mathbf{\Pi},\lambda)$ and $\kappa_{\mathrm{DRP}}(\mathbf{X},\mathbf{R},\lambda)$.

Theorem 28. *If the sketching matrices* $\mathbf{\Pi} \in \mathbb{R}^{n\times m}$ *and* $\mathbf{R} \in \mathbb{R}^{p\times d}$ *are sampled from sub-Gaussian distributions, and satisfying* $\mathbb{E}\left[\mathbf{R}\mathbf{R}^\top\right] = \mathbf{I}_p$ *and* $\mathbb{E}\left[\mathbf{\Pi}\mathbf{\Pi}^\top\right] = \mathbf{I}_n$ *respectively, then*

$$\kappa_{\text{HS}}(\mathbf{X}, \mathbf{\Pi}, \lambda) \leq \left(1 - 2C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}\right)^{-1}$$

and

$$\kappa_{\text{DRP}}(\mathbf{X}, \mathbf{R}, \lambda) \leq \left(1 - 2C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1})}{d}\log\left(\frac{1}{\delta}\right)}\right)^{-1}$$

with probability at least $1-\delta$.

Proof is provided in Appendix 4.6.3. With Theorem 28, we immediately obtain the following corollary, which states the overall convergence of Acc-IHS and Acc-IDRP.

Corollary 29. *Suppose conditions of Theorem 28 hold. If the number of iterations of Acc-IHS satisfy*

$$t \geq \left\lceil \left(1 - 2C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}\right)^{-1/2} \log\left(\frac{2||\mathbf{w}^*||_{\mathbf{X}}}{\epsilon}\right)\right\rceil,$$

then with probability at least $1-\delta$, *we have*

$$||\widehat{\mathbf{w}}_{\text{HS}}^{(t)} - \mathbf{w}^*||_{\mathbf{X}} \leq \epsilon.$$

If the number of iterations of Acc-IDRP satisfies

$$t \geq \left\lceil \left(1 - 2C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1})}{d}\log\left(\frac{1}{\delta}\right)}\right)^{-1/2} \log\left(\frac{2||\mathbf{w}^*||_2}{\epsilon}\right)\right\rceil$$

then with probability at least $1-\delta$, *we have*

$$||\widehat{\mathbf{w}}_{\mathrm{DRP}}^{(t)} - \mathbf{w}^*||_2 \leq \epsilon.$$

We can compare the convergence rate of Acc-IHS and Acc-IDRP with that of the standard IHS [Pilanci and Wainwright, 2016] and the IDRP [Zhang et al., 2014].

Remark 5. *The number of iterations to reach* ϵ*-accuracy for IHS is* $\mathcal{O}\left(\left(\frac{1+\rho}{1-\rho}\right)\log\left(\frac{2||\mathbf{w}^*||_{\mathbf{X}}}{\epsilon}\right)\right)$, *where* $\rho = C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}$ *[Corollary 1 Pilanci and Wainwright, 2016]. Acc-IHS reduces the number of iterations to* $\mathcal{O}\left(\left(\sqrt{\frac{1}{1-2\rho}}\right)\log\left(\frac{2||\mathbf{w}^*||_{\mathbf{X}}}{\epsilon}\right)\right)$, *which is significantly smaller when* ρ *is relatively large. Furthermore, IHS requires* $m \gtrsim \mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})$ *for the convergence to happen, while Acc-IHS is always guaranteed to converge. This will be illustrated in simulations.*

Remark 6. *In a setting with low-rank data, [Theorem 7 Zhang et al., 2014] showed that IDRP requires* $\mathcal{O}\left(\frac{1+\rho}{1-\rho}\right)\log\left(\frac{2||\mathbf{w}^*||_2}{\epsilon}\right)$ *to* ϵ*-accuracy where* $\rho = C_0\sqrt{\frac{r}{d}\log\left(\frac{r}{\delta}\right)}$. *Acc-IDRP reduces the number of iterations to* $\mathcal{O}\left(\sqrt{\frac{1}{1-2\rho}}\right)\log\left(\frac{2||\mathbf{w}^*||_2}{\epsilon}\right)$ *and, furthermore, relaxes the stringent condition* $d \gtrsim r\log r$ *needed for IDRP to converge, since Acc-IDRP always converges.*

4.3.3 Analysis of the primal-dual sketch method

In this section, we provide analysis for the primal-dual sketch method. Recall that here the sketched dual problem is not solved exactly, but is approximately solved by sketching the primal problem again.

Consider an outer loop iteration t. The analysis of the iterative Hessian sketch gives us the following lemma on the decrease of the error.

Lemma 30. *Let $\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}$ be the iterate defined in Algorithm 4. Then we have*

$$||\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \mathbf{w}^*||_{\mathbf{X}} \leq \frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}}\log\left(\frac{1}{\delta}\right)}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}}\log\left(\frac{1}{\delta}\right)}||\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)} - \mathbf{w}^*||_{\mathbf{X}}.$$

Note, however, that in the iterative primal-dual sketch, we do not have access to the exact minimizer $\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}$. Instead, we have an *approximate* minimizer $\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}$, which is close to $\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}$. So it remains to analyze the iteration complexity of the inner loop and see how close the approximate minimizer $\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}$ is to the optimal solution $\mathbf{w}^*$. We have the following theorem.

Theorem 31. *With probability at least $1 - \delta$, we have the following approximation error bound for $\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}$ in the iterative primal-dual sketch:*

$$\begin{aligned}||\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \mathbf{w}^*||_{\mathbf{X}} \leq{}& \left(\frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}}\log\left(\frac{1}{\delta}\right)}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}}\log\left(\frac{1}{\delta}\right)}\right)^t ||\mathbf{w}^*||_{\mathbf{X}} \\ &+ \frac{10\lambda_{\max}^2\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)}{\lambda^2}\left(\frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}\right)^k ||\mathbf{w}^*||_2.\end{aligned}$$

The proof is given in Appendix 4.6.5.

With Theorem 31, we have the following iterative complexity for the proposed IPDS approach.

Corollary 32. *If the number of outer iterations t and number of inner iterations k in the*

IPDS satisfy

$$t \geq \left\lceil \frac{1 + C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}} \right\rceil \log\left(\frac{4\|\mathbf{w}^*\|_{\mathbf{X}}}{\epsilon}\right),$$

$$k \geq \left\lceil \frac{1 + C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1})}{d}\log\left(\frac{1}{\delta}\right)}}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1})}{d}\log\left(\frac{1}{\delta}\right)}} \right\rceil \log\left(\frac{40\lambda_{\max}^2\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)\|\mathbf{w}^*\|_2}{\lambda\epsilon}\right),$$

then with probability at least $1-\delta$*:*

$$\|\widetilde{\mathbf{w}}_{\text{IPDS}}^{(t+1)} - \mathbf{w}^*\|_{\mathbf{X}} \leq \epsilon.$$

Proof. The result directly follows by an application of Theorem 31. □

Remark 7. *The total number of sketched subproblem to solve in iterative primal-dual sketch is* $t \cdot k$*. To obtain* ϵ *approximation error, the total number of subproblems is*

$$tk \lesssim \left\lceil \frac{1 + \sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}}}{1 - \sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}}} \cdot \frac{1 + \sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1})}{d}}}{1 - \sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1})}{d}}} \right\rceil \log^2\left(\frac{1}{\epsilon}\right).$$

Thus the iterative primal-dual sketch will be efficient when the Gaussian width of set $\mathbf{X}\mathbb{R}^p$ *and* $\mathbf{X}^\top\mathbb{R}^n$ *is relatively small. For example, when* $\operatorname{rank}(\mathbf{X}) = r \ll \min(n,p)$*, we can choose the sketching dimension in IPDS to be* $m, d \gtrsim r$*. In this case the IPDS can return a solution with* ϵ*-approximation error by solving* $\log^2(1/\epsilon)$ *small scale subproblems of scale* $r \times r$*.*

We next provide iteration complexity for the proposed Acc-IPDS algorithms in Algorithm 10.

Corollary 33. *If the number of outer loops t and number of inner loops k in IPDS satisfy*

$$t \geq \left\lceil \left(1 - 2C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}} \log\left(\frac{1}{\delta}\right)\right)^{-1/2} \right\rceil \log\left(\frac{4||\mathbf{w}^*||_{\mathbf{X}}}{\epsilon}\right),$$

$$k \geq \left\lceil \left(1 - 2C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1})}{d}} \log\left(\frac{1}{\delta}\right)\right)^{-1/2} \right\rceil \log\left(\frac{40\lambda_{\max}^2\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)||\mathbf{w}^*||_2}{\lambda\epsilon}\right),$$

then with probability at least $1-\delta$*:*

$$||\widetilde{\mathbf{w}}_{\text{IPDS}}^{(t+1)} - \mathbf{w}^*||_{\mathbf{X}} \leq \epsilon.$$

Proof. The proof is similar to the proof of Theorem 31. We simply need to substitute the lower bounds for t and k to obtain the desired result. □

4.3.4 Runtime comparison for large n, large p, and low-rank data

To solve problem (4.1), the runtime usually depends on several quantities including the sample size n, the dimension p, as well as the condition number. To make the comparison between different algorithms, we simply assume $\mathbf{X}$ is of rank r, noting that r might be much smaller than n and p. In (4.1), the regularization parameter λ is generally chosen at the order of $\mathcal{O}(1/\sqrt{n})$ to $\mathcal{O}(1/n)$ [Sridharan et al., 2009, Dhillon et al., 2013]. Here, we simply consider the large value for λ, that is, of order $\mathcal{O}(1/\sqrt{n})$, which gives a better condition number for the problem. For iterative optimization algorithms, the convergence depends on the smoothness parameter of the problem. In (4.1), the smoothness parameter is $\lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p\right)$, which is often of the order $\mathcal{O}(p)$, for example, under a random sub-Gaussian design. To attain the runtime of solving (4.1) in different scenarios, we consider the following methods which are summarized in Table 4.2 with their time complexities in terms of stated parameters:

Solving Linear System: which solves the problem exactly using matrix inversion, and

Approach / Runtime	$\mathcal{O}(\cdot)$	Comment
`Linear System`	$np^2 + p^3$	
`LS with Low-rank SVD`	$npr + r^3$	
`Gradient Descent`	$\left(n^{1.5}p^2\right) \log\left(\frac{1}{\varepsilon}\right)$	
`Acc.Gradient Descent`	$\left(n^{1.25}p^{1.5}\right) \log\left(\frac{1}{\varepsilon}\right)$	
`Coordinate Descent`	$\left(n^{1.5}p\right) \log\left(\frac{1}{\varepsilon}\right)$	
`SVRG,SDCA,SAG`	$\left(np + n^{0.5}p^2\right) \log\left(\frac{1}{\varepsilon}\right)$	
`Catalyst,APPA`	$\left(np + n^{0.75}p^{1.5}\right) \log\left(\frac{1}{\varepsilon}\right)$	
`DSPDC`	$npr + \left(nr + n^{0.75}p^{1.5}r\right) \log\left(\frac{1}{\varepsilon}\right)$	
`IHS + Catalyst`	$np \log p + n^{0.25}p^{1.5}r \log^2\left(\frac{1}{\varepsilon}\right)$	Fast when $p \ll n$
`DRP + Exact`	$np \log n + (nr^2 + r^3) \log\left(\frac{1}{\varepsilon}\right)$	Fast when $n \ll p$
`Iter.primal-dual sketch`	$np \log p + (n + r^3) \log^2\left(\frac{1}{\varepsilon}\right)$	Fast when $r \ll \max(p, n)$

Table 4.2: Comparison of runtime of different approaches for solving the large scale optimization problem in (4.1) stated in terms of number of samples n, the dimensionality of data points p, the rank of data matrix r, and the target accuracy of recovered solution ε.

requires $\mathcal{O}(np^2 + p^3)$.

Linear System with Low-rank SVD: if we have the factorization $\mathbf{X} = \mathbf{U}\mathbf{V}^\top$ available, where $\mathbf{U} \in \mathbb{R}^{n \times r}$, $\mathbf{V} \in \mathbb{R}^{p \times r}$, then we can solve the matrix inversion efficiently using the Sherman-Morrison-Woodbury formula: $\left(\lambda \mathbf{I}_p + \frac{\mathbf{X}^\top \mathbf{X}}{n}\right)^{-1} = \frac{1}{\lambda}\mathbf{I}_p - \frac{1}{\lambda^2}\mathbf{V}\mathbf{U}^\top\mathbf{U}(\mathbf{I}_r + \mathbf{V}^\top\mathbf{V}\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{V}^\top$. This can be done in $\mathcal{O}(npr + r^3)$ in total.

Gradient Descent: standard analysis [Nesterov, 2013] shows that the gradient descent requires $\mathcal{O}\left(\left(\frac{L}{\lambda}\right) \log\left(\frac{1}{\varepsilon}\right)\right)$ iterations, with each iteration has a time complexity of $\mathcal{O}(np)$ to compute the full gradient for all training samples. Since $L = \mathcal{O}(p), \lambda = \mathcal{O}\left(1/\sqrt{n}\right)$, the overall runtime becomes $\mathcal{O}\left(\left(n^{1.5}p^2\right) \log\left(\frac{1}{\varepsilon}\right)\right)$.

Accelerated Gradient Descent [Nesterov, 2013]: which requires $\mathcal{O}\left(\sqrt{\left(\frac{L}{\lambda}\right)} \log\left(\frac{1}{\varepsilon}\right)\right)$ iterations, where the cost of each iteration is $\mathcal{O}(np)$. For the stated values of parameters $L = \mathcal{O}(p)$ and $\lambda = \mathcal{O}\left(1/\sqrt{n}\right)$, the overall runtime would be $\mathcal{O}\left(\left(n^{1.25}p^{1.5}\right) \log\left(\frac{1}{\varepsilon}\right)\right)$.

Randomized Coordinate Descent [Nesterov, 2012]: which requires $\mathcal{O}\left(p\left(\frac{1}{\lambda}\right) \log\left(\frac{1}{\varepsilon}\right)\right)$ iterations, with each iteration $\mathcal{O}(n)$, since $\lambda = \mathcal{O}\left(1/\sqrt{n}\right)$. We have the overall runtime is $\mathcal{O}\left(\left(n^{1.5}p\right) \log\left(\frac{1}{\varepsilon}\right)\right)$.

SVRG, SDCA, SAG [Johnson and Zhang, 2013, Zhang et al., 2013a, Shalev-Shwartz and Zhang, 2013, Roux et al., 2012]: which requires $\mathcal{O}\left(\left(n+\frac{L}{\lambda}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$ iterations, with the time complexity of $\mathcal{O}(p)$ for each iteration to computed the gradient of simple sample. Since $L=\mathcal{O}(p), \lambda=\mathcal{O}\left(1/\sqrt{n}\right)$, the overall runtime for this family of algorithms would be $\mathcal{O}\left(\left(np+n^{0.5}p^2\right)\log\left(\frac{1}{\varepsilon}\right)\right)$.

Accelerated SVRG: Catalyst, APPA, SPDC, RPDG [Lin et al., 2015, Frostig et al., 2015, Zhang and Xiao, 2017, Lan and Zhou, 2017]: thanks to acceleration, this algorithm requires $\mathcal{O}\left(\left(n+\sqrt{n\frac{L}{\lambda}}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$ iterations, with each iteration shares the same $\mathcal{O}(p)$ complexity per iteration as SVRG. Since $L=\mathcal{O}(p), \lambda=\mathcal{O}\left(1/\sqrt{n}\right)$, the overall runtime becomes
$\mathcal{O}\left(\left(np+n^{0.75}p^{1.5}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$.

DSPDC [Yu et al., 2015]: requires $\mathcal{O}\left(\left(n+\sqrt{n\frac{L}{\lambda}p}\right)\log\left(\frac{1}{\varepsilon}\right)\right)$ iterations, and each iteration is in order of $\mathcal{O}(r)$. Here $L=\mathcal{O}(p), \lambda=\mathcal{O}\left(1/\sqrt{n}\right)$. Also, to apply DSPDC, one should compute the low-rank factorization as a preprocessing step which takes $\mathcal{O}(npr)$. Thus we have the overall runtime for this algorithm as $\mathcal{O}\left(npr+\left(nr+n^{0.75}p^{0.5}r\right)\log\left(\frac{1}{\varepsilon}\right)\right)$.

Iterative Hessian Sketch + Accelerated SVRG [Pilanci and Wainwright, 2016]: computing the sketched problem takes $\mathcal{O}(np\log p)$ (e.g., via fast Johnson-Lindenstrauss transforms [Ailon and Chazelle, 2009]). The algorithms solves $\mathcal{O}\left(\log\left(\frac{1}{\varepsilon}\right)\right)$ sketched problems using accelerated SVRG type algorithm that takes $\mathcal{O}\left(n^{0.25}p^{1.5}r\log\left(\frac{1}{\varepsilon}\right)\right)$. This leads to the overall runtime of $\mathcal{O}\left(np\log p+n^{0.25}p^{1.5}r\log^2\left(\frac{1}{\varepsilon}\right)\right)$.

DRP + Matrix inversion [Zhang et al., 2014]: computing the sketched problem takes $\mathcal{O}(np\log n)$. The algorithms needs to solve $\mathcal{O}\left(\log\left(\frac{1}{\varepsilon}\right)\right)$ reduced problems where each of them requires a matrix inversion with time complexity of $\mathcal{O}\left(nr^2+r^3\right)$. This leads to the overall runtime of $\mathcal{O}\left(np\log n+(nr^2+r^3)\log\left(\frac{1}{\varepsilon}\right)\right)$ for this algorithm.

Iterative Primal-Dual Sketch: computing the sketched problem takes $\mathcal{O}(np\log p)$. The algorithms iterates for $\mathcal{O}\left(\log^2\left(\frac{1}{\varepsilon}\right)\right)$ rounds, and at each iteration it needs to solve a reduced problem that exactly takes $\mathcal{O}\left(n+r^3\right)$. As a result the overall runtime becomes as

$\mathcal{O}\left(np\log p + (n + r^3)\log^2\left(\frac{1}{\varepsilon}\right)\right)$.

4.4 Communication-efficient distributed optimization via sketching

In this section we apply the improved iterative sketching in the distributed optimization problems. Typically, distributed optimization approaches can be divided into two categories, depending how the data set is partitioned across different machines: data could be partitioned across features [Heinze et al., 2014, 2015, Wang et al., 2016c, Yang et al., 2016] or it could be partitioned by samples [Shamir et al., 2014, Zhang and Xiao, 2015, Lee et al., 2017b, Wang et al., 2017a, Jordan et al., 2018, Smith et al., 2016]. For the setting where features are partitioned across machines, we propose the (accelerated) iterative distributed dual random projection (DIDRP). In the setting where samples are partitioned across machines, we propose the (accelerated) iterative distributed Hessian sketch (DIHS). We discuss in detail how these proposals compare to and improve over existing work.

4.4.1 Distributed iterative dual random projection

We first consider a setting where features are distributed across different machines. In this setting, LOCO [Heinze et al., 2014] and Dual-LOCO [Heinze et al., 2015] considered sketching based approaches, where randomly projected data are transmitted across machines to approximate the original data. However, as predicted by theory, these one-shot approaches require communicating a very large number of vectors in order to obtain a high accuracy solution for the original optimization problem. On the other hand, iterative sketching methods are very powerful in reducing the approximation error by solving a different problem using the same sketched data. At the same time, once we have transmitted the sketched data matrix, at every iterative sketching round each machine only needs to communicate two vectors in $\mathbb{R}^n$ to solve the next sketched problem.

Suppose $\mathbf{X} \in \mathbb{R}^{n\times d}$ is partitioned across features over m machines, $\mathbf{X} = [\mathbf{X}_{[1]}, \mathbf{X}_{[2]}, \ldots, \mathbf{X}_{[m]}]$, such that machine k holds $\mathbf{X}_{[k]}$ consisting of p/m features (for simplicity we assume that m divides p). Without loss of generality, assume the first machine serves as the master machine, and it contains the local data $\mathbf{X}_{[1]}$, as well as the transmitted, randomly projected data $[\mathbf{X}_{[2]}\mathbf{R}_{[2]}, \mathbf{X}_{[3]}\mathbf{R}_{[3]}, \ldots, \mathbf{X}_{[m]}\mathbf{R}_{[m]}]$. Let $\widetilde{\mathbf{X}} = [\mathbf{X}_{[1]}, \mathbf{X}_{[2]}\mathbf{R}_{[2]}, \mathbf{X}_{[3]}\mathbf{R}_{[3]}, \ldots, \mathbf{X}_{[m]}\mathbf{R}_{[m]}]$ be the concatenated data matrix which contains full local data and the sketched global data. Here each random matrix $\mathbf{R}_{[k]}$, $k = 2, \ldots, m$ is of dimension $(p/m) \times (d/(m-1))$, so that the dimension of $\widetilde{\mathbf{X}}$ is $n \times (p/m) + d$. At each iteration, the master machine solves the following problem:

$$\min_{\mathbf{z}} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}^{(t)} - \widetilde{\mathbf{X}}\mathbf{z}||_2^2 + \frac{\lambda}{2}||\mathbf{z} + \widehat{\mathbf{w}}^{(t)}||_2^2, \tag{4.34}$$

where $\widehat{\mathbf{w}}^{(t)} = [\mathbf{w}_{[1]}^{(t)}; \mathbf{R}_{[2]}^\top \mathbf{w}_{[2]}^{(t)}; \ldots; \mathbf{R}_{[m]}^\top \mathbf{w}_{[m]}^{(t)}]$. In order to do so, each machine communicates $\mathbf{X}_{[k]}\widehat{\mathbf{w}}_{[k]}^{(t)}$, and the master machine aggregates and computes $\mathbf{X}\widehat{\mathbf{w}}^{(t)} = \sum_{k=1}^m \mathbf{X}_{[k]}\widehat{\mathbf{w}}_{[k]}^{(t)}$. With $\widehat{\mathbf{z}}$ obtained by solving (4.34), the master machine can update the dual solution $\widehat{\boldsymbol{\alpha}}^{(t+1)}$ and communicate it back to each machine. Each machine, in turn, uses the obtained $\widehat{\boldsymbol{\alpha}}^{(t+1)}$ to updated their local primal solution as $\widehat{\mathbf{w}}_{[k]}^{(t+1)} = \frac{1}{\lambda n} X_{[k]}^\top \widehat{\boldsymbol{\alpha}}^{(t+1)}$. The details of the algorithm are presented in Algorithm 11. It is noteworthy to point out that after the initial transmission stage, in each iteration, each worker only communicates two vectors in $\mathbb{R}^n$ to the master machine.

The following corollary states the communication complexity of Algorithm 11 which is a direct consequence of Theorem 24.

Corollary 34. *Suppose that sub-Gaussian sketching matrices were used in Algorithm 11. For Algorithm 11 to reach ϵ accuracy, $||\widehat{\mathbf{w}}^{(t)} - \mathbf{w}^*||_2 \leq \epsilon$, the total number of vectors (in $\mathbb{R}^n$)*

Algorithm 11 Distributed Iterative Dual Random Projection (DIDRP).

```
Input: Data $\mathbf{X}, \mathbf{y}$.
Initialization: $\widehat{\mathbf{w}}^{(0)} = \mathbf{0}$.
for Each worker $k = 2, ..., m$ do
    Compute and communicate randomly projected data $\mathbf{X}_{[k]}\mathbf{R}_{[k]}$.
end
for $t = 0, 1, 2, \ldots$ do
    The master machine solves the projected problem in (4.34), and obtains $\widehat{\mathbf{z}}^{(t)}$.
    The master machine computes and communicates the dual approximation: $\widehat{\boldsymbol{\alpha}}^{(t+1)} = \mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}^{(t)} - \widetilde{\mathbf{X}}\widehat{\mathbf{z}}^{(t)}$.
    for Each worker $k = 2, ..., m$ do
        Update local primal approximation: $\widehat{\mathbf{w}}_{[k]}^{(t+1)} = \frac{1}{\lambda n}\mathbf{X}_{[k]}^{\top}\widehat{\boldsymbol{\alpha}}^{(t+1)}$.
        Compute and communicate $\mathbf{X}_{[k]}\widehat{\mathbf{w}}_{[k]}^{(t+1)}$.
    end
end
```

each machine needs to communicate is upper bounded by

$$\mathcal{O}\left(\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p)}{m-1} + \log\left(\frac{||\mathbf{w}^*||_2}{\epsilon}\right)\right).$$

Remark 8. *We can compare our result with that established for Dual-LOCO [Heinze et al., 2015]. Dual-LOCO requires the number of communication rounds to linearly with* $1/\epsilon^2$*. On the other hand, the number of communication rounds of DIDRP only grow logarithmically with* $1/\epsilon$*. Therefore, DIDRP presents a significant improvement over Dual-LOCO. This is also verified by the empirical results.*

4.4.2 Distributed iterative Hessian sketch

Next, we consider a setting where the data are partitioned by samples. The data matrix $\mathbf{X}$ is partitioned as $\mathbf{X} = [\mathbf{X}_{(1)}; \mathbf{X}_{(2)}; \ldots; \mathbf{X}_{(m)}]$ where each machine k holds the local data $\mathbf{X}_{(k)} \in \mathbb{R}^{n/m \times p}$, which contains n/m samples. In this setting, our main idea is to approximate the Hessian matrix with a mix of local data and sketched global data. Again, assume the first machine serves as the master. At the beginning of the algorithm, workers compute and communicate their sketched local data $\mathbf{\Pi}_{(k)}^\top \mathbf{X}_{(k)}$. The master constructs the sketched data matrix as $\widetilde{\mathbf{X}} = [\mathbf{X}_{(1)}; \mathbf{\Pi}_{(2)}^\top \mathbf{X}_{(2)}; \ldots; \mathbf{\Pi}_{(m)}^\top \mathbf{X}_{(m)}]$, which will be used for constructing an approximation to the Hessian matrix as $\widetilde{\mathbf{H}} = \frac{\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}}}{n/m+d}$. At each iteration of the algorithm, the master solves a sub-problem of form

$$\widehat{\mathbf{u}}^{(t)} = \arg\min_{\mathbf{u}} \mathbf{u}^\top \left(\widetilde{\mathbf{H}} + \frac{\lambda}{2} \mathbf{I}_p \right) \mathbf{u} - \left\langle \frac{\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}^{(t)})}{n} - \lambda \widehat{\mathbf{w}}^{(t)}, \mathbf{u} \right\rangle, \tag{4.35}$$

which is inspired by the iterative Hessian sketch. The quantity $\mathbf{X}^\top \mathbf{X} \widehat{\mathbf{w}}^{(t)}$ is computed by communicating and aggregating the local information $\mathbf{X}^\top \mathbf{X} \widehat{\mathbf{w}}^{(t)} = \sum_{k=1}^m \mathbf{X}_{(k)}^\top \mathbf{X}_{(k)} \widehat{\mathbf{w}}^{(t)}$. The details of the algorithm DIHS are presented in Algorithm 12. The following corollary on its communication efficiency is a direct consequence of Theorem 24.

Corollary 35. *Suppose we use sub-Gaussian sketching in Algorithm 11 and for Algorithm 12 to reach ϵ approximation: $||\widehat{\mathbf{w}}^{(t)} - \mathbf{w}^*||_{\mathbf{X}} \leq \epsilon$, the total number of vectors (in $\mathbb{R}^p$) each machine need to communicate is upper bounded by*

$$\mathcal{O}\left(\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p)}{m-1} + \log\left(\frac{||\mathbf{w}^*||_{\mathbf{X}}}{\epsilon} \right) \right).$$

Acceleration The acceleration techniques presented in Section 4.2.1 and 4.2.3 can also be applied in the distributed optimization setting to further improve the communication efficiency of DIDRP and DIHS. In the experiments, we found that the accelerated algorithms can often help in saving communication because of their faster convergence.

4.5 Experiments

In this section we present extensive comparisons for the proposed iterative sketching approaches on both simulated and real world data sets. We first demonstrate the improved

Algorithm 12 Distributed Iterative Hessian Sketch (DIHS).

```
Input: Data $\mathbf{X}, \mathbf{y}$.
Initialization: $\widehat{\mathbf{w}}^{(0)} = \mathbf{0}$.
for Each work $k = 2, ..., m$ do
    Compute and communicate randomly projected data $\mathbf{\Pi}_{(k)}^\top \mathbf{X}_{(k)}$.
end
for $t = 0, 1, 2, \ldots$ do
    for Each worker $k = 2, ..., m$ do
        Compute and communicate $\mathbf{X}_{(k)}^\top \mathbf{X}_{(k)} \widehat{\mathbf{w}}^{(t)}$.
    end
    The master machine computes $\widehat{\mathbf{w}}^{(t+1)} = \widehat{\mathbf{w}}^{(t)} + \widehat{\mathbf{u}}^{(t)}$, where $\widehat{\mathbf{u}}^{(t)}$ is obtained by solving
    the sketched problem (4.35), and communicates $\widehat{\mathbf{w}}^{(t+1)}$.
end
```

convergence of the proposed Acc-IHS and Acc-IDRP algorithms on simulated data sets. Then we show that the proposed iterative primal-dual sketch procedure and its accelerated version could simultaneously reduce the sample size and dimension of the problem, while still maintaining high approximation precision. Finally, we evaluate these algorithms on some real world data sets.

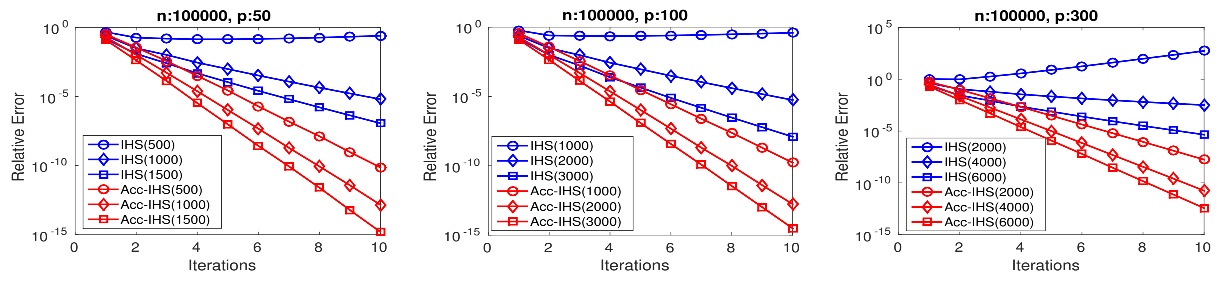
4.5.1 Simulations for Acc-IHS and Acc-IDRP

We first examine the effectiveness of the proposed Acc-IHS and Acc-DRP algorithms on simulated data. We generate the response $\{y_i\}_{i\in[n]}$ from the following linear model

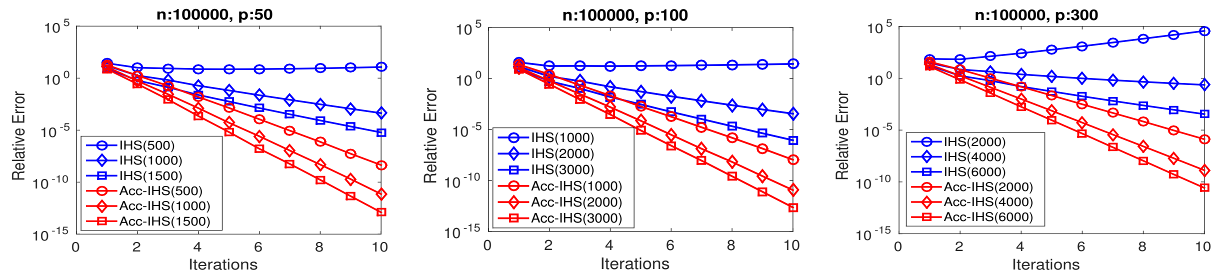
$$y_i = \langle \mathbf{x}_i, \mathbf{w}^* \rangle + \epsilon_i,$$

where the noise ϵ_i is sampled from a standard Normal distribution. The true model $\mathbf{w}^*$ is a p-dimensional vector where the entries are sampled i.i.d. from a uniform distribution in $[0, 1]$.

We first compare the proposed Acc-IHS with the standard IHS on some "big n", but relatively low-dimensional problems. We generate $\{\mathbf{x}_i\}_{i\in[n]}$ from a multivariate Normal



$$\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \Sigma_{ij} = 0.5^{|i-j|}.$$



$$\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \Sigma_{ij} = 0.5^{|i-j|/10}.$$

Figure 4.1: Comparison of IHS and Acc-IHS on various simulated datasets. The sketching dimension for each algorithm is shown inside parentheses.

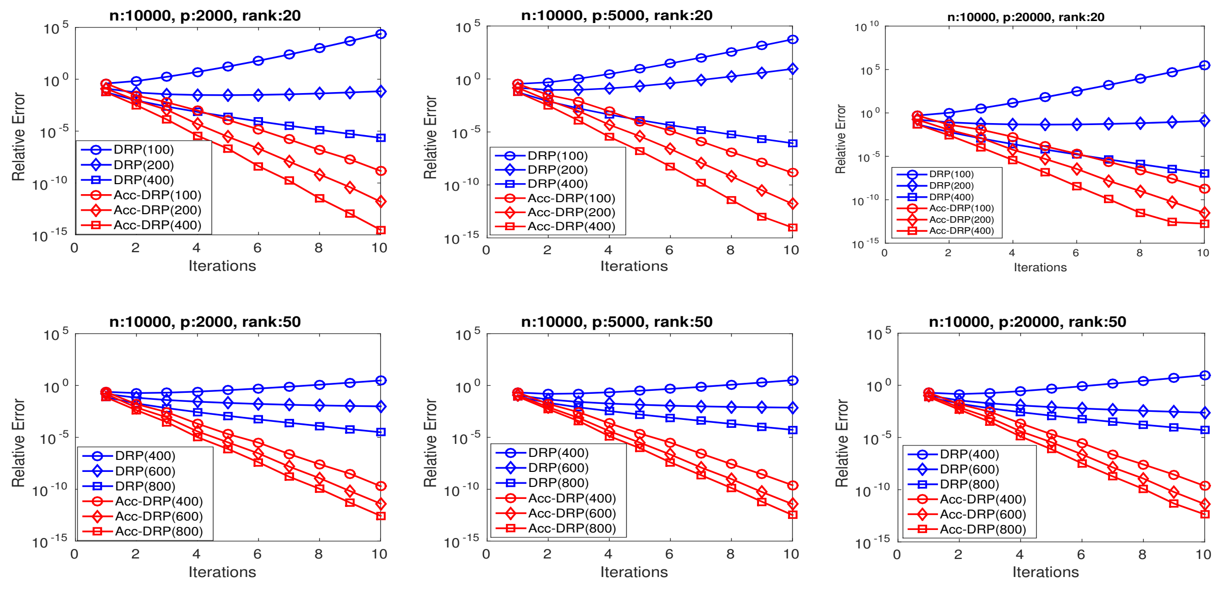


Figure 4.2: Comparison of IDRP and Acc-IDRP on various simulated datasets.

Table 4.3: List of real-world data sets used in the experiments.

Name	#Instances	#Features
connect4	67,557	126
slice	53,500	385
year	51,630	90
colon-cancer	62	2,000
duke breast-cancer	44	7,129
leukemia	72	7,129
cifar	4,047	3,072
gisette	6,000	5,000
sector	6,412	15,000
mnist	60,000	780
tomes	28,179	96
twitter	582,350	77

distribution with mean zero vector, and covariance matrix $\mathbf{\Sigma}$, which controls the condition number of the problem. We will varying $\mathbf{\Sigma}$ to see how it affects the performance of various methods. We set $\Sigma_{ij} = 0.5^{|i-j|}$ for the well-conditioned setting, and $\Sigma_{ij} = 0.5^{|i-j|/10}$ for the ill-conditioned setting. We fix the sample size $n = 10^5$ and vary the dimension $p \in \{50, 100, 300\}$. The results are shown in Figure 4.1. For each problem setting, we test 3 different sketching dimensions (number inside parentheses in legend). We have the following observations:

- For both IHS and Acc-IHS, the larger the sketching dimension m, the faster the iterative algorithms converges to the optimum, which is consistent with the theory that characterize the benefit of using larger sketching dimension. And this has also been observed in [Pilanci and Wainwright, 2016] and [Zhang et al., 2014] for IHS and IDRP algorithms.

- When compared with IHS, we observe that Acc-IHS converges significantly faster. Moreover, when the sketching dimension is small, IHS can diverge and go far away from the optimum, while Acc-IHS still converges.

- For all the simulation setting we tried, Acc-IHS converges faster than IHS, even when its sketching dimension is only 1/3 of the sketching dimension of IHS.

Next, we compare the proposed Acc-IDRP with the standard IDRP on high-dimensional, but relatively low-rank data. We generate $\{\mathbf{x}_i\}_{i\in[n]}$ from a low-rank factorization $\mathbf{X} = \mathbf{U}\mathbf{V}^\top$, where the entries in $\mathbf{U} \in \mathbb{R}^{n\times r}$ and $\mathbf{V} \in \mathbb{R}^{p\times r}$ are sampled i.i.d. from a standard Normal distribution. We fix the sample size $n = 10^4$ and vary the dimensions $p \in \{2000, 5000, 20000\}$. We also vary the rank $r \in \{20, 50\}$. The results are shown in Figure 4.2. For each problem setting, we test 3 different sketching dimensions (number inside parentheses in legend). We have similar observations as in the IHS case. Acc-IDRP always converges significantly faster than IDRP. When the low sketching dimension causes IDRP to diverge, Acc-IDRP still converges to the optimum.

Above simulations validate the theoretical analysis, which showed that the accelerated procedures for IHS and IDRP could significantly boost the convergence speed of their standard counterparts. Since the computational cost per iteration of the standard iterative sketching techniques and their accelerated versions is almost the same, Acc-IHS and Acc-IDRP will be useful practical techniques.

4.5.2 Simulations for IPDS and Acc-IPDS

In this section we demonstrate how iterative primal-dual sketch and its accelerated version work on simulated data. We generated the data using the same procedure described in the previous section for Acc-DRP. We generate the low-rank data matrix $\mathbf{X}$ with rank 10 and vary the sample size n and dimension p. For primal-dual sketching, we reduce the sample size to m and the dimension to d, with $m \ll n, d \ll p$. We compare with the standard IHS and IDRP. For IHS, we perform the sample reduction from n to m, while for IDRP we perform data dimension reduction from p to d. Thus the sizes of the sub-problems for IPDS (and Acc-IPDS), IHS, and IDRP are $m \times d$, $m \times p$, and $n \times d$, respectively. For IPDS and

Acc-IPDS, we terminate the inner loop when the ℓ_∞ distance between two inner iterations is less than 10^{-10}. The results are shown in Figure 4.3, where the sketched dimension (m, d) is shown in legend.

We have the following observations:

- IPDS and Acc-IPDS are able to recover the optimum to a very high precision, even though they simultaneously reduce the sample size and data dimension. However, they generally require more iterations to reach certain approximation level compared with IHS and IDRP, which, on the other hand, need to solve a substantially larger subproblem at each iteration. Therefore, primal-dual sketching approach still enjoys computational advantages. For example, on a problem of size $(n, p) = (10000, 20000)$, IHS and IDRP need to solve 5 sub-problems of scale $(m, p) = (500, 20000)$ and $(n, d) = (10000, 500)$, respectively, while Acc-IPDS is only required to solve 35 sub-problems of scale $(m, d) = (500, 500)$ to obtain the same approximation accuracy.

- Acc-IPDS converges significantly faster than IPDS, which again verifies the effectiveness of the proposed acceleration procedure for the sketching techniques.

4.5.3 Experiments on real data sets

In this section, we present experiments conducted on real-world data sets. Table 5.2 summarizes their statistics. Among these data sets, the first 3 are cases where sample size is significantly larger than the data dimension. We use them to compare the IHS and Acc-IHS algorithms. The middle 3 data sets are high-dimensional data sets with small sample sizes. We use them to compare the DRP and Acc-DRP algorithms. Finally, the last 3 data sets are cases where the sample size and data dimensions are both relatively large, which is suitable for iterative primal-dual sketching methods. For the last 3 data sets, we found that standard IHS and DRP often fail, unless a very large sketching dimension is used. As a result, we compared with Acc-IHS and Acc-DRP algorithms. We follow the same experimental setup used in the simulation study. The convergence plots are summarized in Figure 4.4.

We have the following observations:

- Acc-IHS and Acc-DRP converge significantly faster than IHS and DRP, respectively. This is consistent with the observation drawn from simulation studies.

- For the last 3 data sets, where n and p are both large, and the data are not exactly low-rank, IHS, DRP, and IPDS often diverge. This is because the requirement on the sketching dimension to ensure convergence is high. The accelerated versions still converge to the optimum. It is notable that the Acc-IPDS only requires solving several least squares problems with both sample size and dimension being relatively small.

4.5.4 Experiments for distributed optimization

We consider distributed optimization in both partition by features and partition by samples settings. We follow the data generating process as in the simulation study for Acc-DRP, where we fix a low-rank data matrix $\mathbf{X}$ with rank 10, and vary the sample size n and dimension p, as well as number of machines m. For partition by features scenario, we compare with LOCO and Dual-LOCO [Heinze et al., 2014, 2015]. We plot the relative approximation error versus the number of vectors (in $\mathbb{R}^n$) communicated. The results are shown in Figure 4.5. We LOCO and Dual-LOCO fail to quickly decrease the approximation error even with relatively large communication, this is consistent with theory that characterize the limit of one-shot sketching methods. The proposed DIDRP algorithm clearly outperforms LOCO methods as the number of communicated vectors grows. We further observe that Acc-DIDRP is more efficient than DIDRP, which again illustrates that the acceleration techniques can be helpful in further reducing the communication.

For the partition by sample scenario, we compare with several state-of-the-art algorithms including accelerated gradient descent (AccGD) [Nesterov, 2013], ADMM [Boyd et al., 2011], DANE [Shamir et al., 2014] and DiSCO [Zhang and Xiao, 2015]. The results are summarized in Figure 4.6. We observe that the methods leveraging higher-order information (DANE,DiSCO,DIHS,Acc-DIHS) are significantly more communication-efficient compared to AccGD and ADMM. Generally speaking Acc-DIHS has a slight advantage over existing approaches. We also tested on several real world data sets and the results are shown in Figure 4.7, where we observed a similar behavior.

4.6 Proofs of technical results

The appendix contains proofs of theorems stated in this chapter.

4.6.1 Proof of Theorem 24

Proof. Based on the optimality condition for $\mathbf{w}^*$ and $\widehat{\mathbf{w}}_{\mathrm{HS}}$, we have

$$\left(\frac{\mathbf{X}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p\right)\mathbf{w}^* = \frac{\mathbf{X}^\top\mathbf{y}}{n} \quad \text{and} \quad \left(\frac{\mathbf{X}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p\right)\widehat{\mathbf{w}}_{\mathrm{HS}} = \frac{\mathbf{X}^\top\mathbf{y}}{n}.$$

Therefore

$$\left(\frac{\mathbf{X}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p\right)\mathbf{w}^* - \left(\frac{\mathbf{X}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p\right)\widehat{\mathbf{w}}_{\mathrm{HS}} = \mathbf{0},$$

and

$$\left\langle \left(\frac{\mathbf{X}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p\right)\mathbf{w}^* - \left(\frac{\mathbf{X}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p\right)\widehat{\mathbf{w}}_{\mathrm{HS}}, \mathbf{w}^* - \widehat{\mathbf{w}}_{\mathrm{HS}} \right\rangle = 0.$$

By adding and subtracting $\left\langle \mathbf{w}^* - \widehat{\mathbf{w}}_{\text{HS}}, \left(\frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{n} + \lambda \mathbf{I}_p\right) \mathbf{w}^* \right\rangle$, we have

$$\begin{aligned}
&\left\langle \left(\frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{n} - \frac{\mathbf{X}^\top \mathbf{X}}{n}\right) \mathbf{w}^*, \widehat{\mathbf{w}}_{\text{HS}} - \mathbf{w}^* \right\rangle \\
&= (\mathbf{w}^* - \widehat{\mathbf{w}}_{\text{HS}})^\top \left(\frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{n} + \lambda \mathbf{I}_p\right) (\mathbf{w}^* - \widehat{\mathbf{w}}_{\text{HS}})
\end{aligned}$$

The term on right hand side is lower bounded as

$$\begin{aligned}
(\mathbf{w}^* - \widehat{\mathbf{w}}_{\text{HS}})^\top \left(\frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{n}\right) (\mathbf{w}^* - \widehat{\mathbf{w}}_{\text{HS}}) + \lambda ||\mathbf{w}^* - \widehat{\mathbf{w}}_{\text{HS}}||_2^2 \\
\geq \rho_1(\mathbf{X}\mathbb{R}^p, \mathbf{\Pi}) ||\mathbf{w}^* - \widehat{\mathbf{w}}_{\text{HS}}||_{\mathbf{X}}^2.
\end{aligned} \tag{4.36}$$

For the left hand side, we have the following upper bound

$$\left\langle \left(\mathbf{\Pi}\mathbf{\Pi}^\top - \mathbf{I}_n\right) \frac{\mathbf{X}\mathbf{w}^*}{\sqrt{n}}, \frac{\mathbf{X}}{\sqrt{n}}(\widehat{\mathbf{w}}_{\text{HS}} - \mathbf{w}^*) \right\rangle \leq \rho_2(\mathbf{X}\mathbb{R}^p, \mathbf{\Pi}, \mathbf{w}^*) ||\mathbf{w}^*||_{\mathbf{X}} ||\widehat{\mathbf{w}}_{\text{HS}} - \mathbf{w}^*||_{\mathbf{X}}. \tag{4.37}$$

Combining (4.36) and (4.37) we have

$$||\widehat{\mathbf{w}}_{\text{HS}} - \mathbf{w}^*||_{\mathbf{X}} \leq \frac{\rho_2(\mathbf{X}\mathbb{R}^p, \mathbf{\Pi}, \mathbf{w}^*)}{\rho_1(\mathbf{X}\mathbb{R}^p, \mathbf{\Pi})} ||\mathbf{w}^*||_{\mathbf{X}}.$$

For the recovery of dual variables, we have

$$\begin{aligned}
||\widehat{\boldsymbol{\alpha}}_{\text{HS}} - \boldsymbol{\alpha}^*||_2 =& ||\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}_{\text{HS}} - (\mathbf{y} - \mathbf{X}\mathbf{w}^*)||_2 \\
=& \sqrt{n} ||\widehat{\mathbf{w}}_{\text{HS}} - \mathbf{w}^*||_{\mathbf{X}} \\
\leq& \sqrt{n} \frac{\rho_2(\mathbf{X}\mathbb{R}^p, \mathbf{\Pi}, \mathbf{w}^*)}{\rho_1(\mathbf{X}\mathbb{R}^p, \mathbf{\Pi})} ||\mathbf{w}^*||_{\mathbf{X}}.
\end{aligned}$$

This completes the proof for the Hessian sketch.

For the dual random projection, the proof is mostly analogous. Based on the optimality condition for $\boldsymbol{\alpha}^*$ and $\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}$, we have

$$\left(\frac{\mathbf{X}\mathbf{X}^\top}{n} + \lambda \mathbf{I}_n\right)\boldsymbol{\alpha}^* = \lambda \mathbf{y} \quad \text{and} \quad \left(\frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top\mathbf{X}^\top}{n} + \lambda \mathbf{I}_n\right)\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}} = \lambda \mathbf{y}.$$

Therefore

$$\left(\frac{\mathbf{X}\mathbf{X}^\top}{n} + \lambda \mathbf{I}_n\right)\boldsymbol{\alpha}^* - \left(\frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top\mathbf{X}^\top}{n} + \lambda \mathbf{I}_n\right)\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}} = \mathbf{0},$$

and

$$\left\langle \left(\frac{\mathbf{X}\mathbf{X}^\top}{n} + \lambda \mathbf{I}_n\right)\boldsymbol{\alpha}^* - \left(\frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top\mathbf{X}^\top}{n} + \lambda \mathbf{I}_n\right)\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}, \boldsymbol{\alpha}^* - \widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}} \right\rangle = 0.$$

Simple algebra gives us

$$\begin{aligned}&\left\langle \left(\frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top\mathbf{X}^\top}{n} - \frac{\mathbf{X}\mathbf{X}^\top}{n}\right)\boldsymbol{\alpha}^*, \widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}} - \boldsymbol{\alpha}^* \right\rangle \\ &= (\boldsymbol{\alpha}^* - \widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}})^\top \left(\frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top\mathbf{X}^\top}{n} + \lambda \mathbf{I}_n\right)(\boldsymbol{\alpha}^* - \widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}).\end{aligned}$$

The term on right hand side is lower bounded as

$$\begin{aligned}(\boldsymbol{\alpha}^* - \widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}})^\top \left(\frac{\mathbf{X}\mathbf{R}\mathbf{R}^\top\mathbf{X}^\top}{n}\right)(\boldsymbol{\alpha}^* - \widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}) + \lambda ||\boldsymbol{\alpha}^* - \widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}||_2^2 \\ \geq \rho_1(\mathbf{X}^\top\mathbb{R}^n, \mathbf{R})||\boldsymbol{\alpha}^* - \widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}}||_{\mathbf{X}^\top}^2.\end{aligned} \tag{4.38}$$

The term on the left hand side is upper bounded as

$$\left\langle \left(\mathbf{R}\mathbf{R}^\top - \mathbf{I}_p\right)\frac{\mathbf{X}^\top\boldsymbol{\alpha}^*}{\sqrt{n}}, \frac{\mathbf{X}^\top}{\sqrt{n}}(\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}} - \boldsymbol{\alpha}^*) \right\rangle \leq \rho_2(\mathbf{X}^\top\mathbb{R}^n, \mathbf{R}, \boldsymbol{\alpha}^*)||\boldsymbol{\alpha}^*||_{\mathbf{X}^\top}||\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}} - \boldsymbol{\alpha}^*||_{\mathbf{X}^\top}. \tag{4.39}$$

Combining (4.38) and (4.39) we have

$$||\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}} - \boldsymbol{\alpha}^*||_{\mathbf{X}^\top} \leq \frac{\rho_2(\mathbf{X}^\top\mathbb{R}^n, \mathbf{R}, \boldsymbol{\alpha}^*)}{\rho_1(\mathbf{X}^\top\mathbb{R}^n, \mathbf{R})}||\boldsymbol{\alpha}^*||_{\mathbf{X}^\top}.$$

For the recovery of primal variables, we have

$$\begin{aligned}||\widehat{\mathbf{w}}_{\mathrm{DRP}} - \mathbf{w}^*||_2 =& \frac{1}{\lambda\sqrt{n}}||\widehat{\boldsymbol{\alpha}}_{\mathrm{DRP}} - \boldsymbol{\alpha}^*||_{\mathbf{X}^\top} \leq \frac{1}{\lambda\sqrt{n}}\frac{\rho_2(\mathbf{X}^\top\mathbb{R}^n, \mathbf{R}, \boldsymbol{\alpha}^*)}{\rho_1(\mathbf{X}^\top\mathbb{R}^n, \mathbf{R})}||\boldsymbol{\alpha}^*||_{\mathbf{X}^\top} \\ =& \frac{\rho_2(\mathbf{X}^\top\mathbb{R}^n, \mathbf{R}, \boldsymbol{\alpha}^*)}{\rho_1(\mathbf{X}^\top\mathbb{R}^n, \mathbf{R})}||\mathbf{w}^*||_2.\end{aligned}$$

An application of Lemma 23 concludes the proof. □

4.6.2 Proof of Theorem 25

We only prove the result for the Hessian sketch here as the proof for the dual random projection is analogous. We will make usage of the following concentration result for sub-Gaussian random matrices.

Lemma 36 (Lemma 3 in [Zhang et al., 2014]). *Let $\mathbf{B} \in \mathbb{R}^{r\times m}$ be a random matrix with entries sampled i.i.d. from zero-mean sub-Gaussian distribution with variance $1/m$, then*

$$||\mathbf{B}\mathbf{B}^\top - \mathbf{I}_r||_2 \leq 2\sqrt{\frac{2(r+1)}{m}\log\frac{2r}{\delta}} := \epsilon_1$$

with probability at least $1-\delta$.

Lemma 37 (Theorem 3.2 in [Recht, 2011]). *Let $\mathbf{B} \in \mathbb{R}^{r\times m}$, $\mathbf{A} \in \mathbb{R}^{(n-r)\times m}$ be two random matrices with entries sampled i.i.d. from a zero-mean sub-Gaussian distribution with variance $1/m$, then*

$$||\mathbf{A}\mathbf{B}^\top||_2 \leq \frac{7}{3}\sqrt{\frac{2(n-r)}{m}\log\frac{n}{\delta}} := \tau_1$$

with probability at least $1-\delta$.

Let $\Delta\mathbf{w} = \mathbf{w}^* - \widehat{\mathbf{w}}_{\mathrm{HS}}$. Then

$$\begin{aligned}
||\Delta\mathbf{w}||_{\mathbf{X}}^2 &= \left|\left|\frac{\mathbf{X}\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2 \\
&= \left|\left|\frac{(\mathbf{U}\boldsymbol{\Sigma}_r\mathbf{V}^\top + \mathbf{U}\boldsymbol{\Sigma}_{\bar{r}}\mathbf{V}^\top)\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2 \\
&= \left|\left|\frac{\boldsymbol{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2 + \left|\left|\frac{\boldsymbol{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2
\end{aligned}$$

Consider the term $\Delta\mathbf{w}^\top\left(\frac{\mathbf{X}^\top\boldsymbol{\Pi}\boldsymbol{\Pi}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p\right)\Delta\mathbf{w}$. We have

$$\begin{aligned}
&\Delta\mathbf{w}^\top\left(\frac{\mathbf{X}^\top\boldsymbol{\Pi}\boldsymbol{\Pi}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p\right)\Delta\mathbf{w} \\
&\quad\geq \Delta\mathbf{w}^\top\left(\frac{\mathbf{V}^\top\boldsymbol{\Sigma}_r\mathbf{U}^\top\boldsymbol{\Pi}\boldsymbol{\Pi}^\top\mathbf{U}\boldsymbol{\Sigma}_r\mathbf{V}^\top}{n}\right)\Delta\mathbf{w} + \lambda||\Delta\mathbf{w}||_2^2 \\
&\qquad + 2\Delta\mathbf{w}^\top\left(\frac{\mathbf{V}^\top\boldsymbol{\Sigma}_{\bar{r}}\mathbf{U}^\top\boldsymbol{\Pi}\boldsymbol{\Pi}^\top\mathbf{U}\boldsymbol{\Sigma}_r\mathbf{V}^\top}{n}\right)\Delta\mathbf{w}.
\end{aligned}$$

Since

$$\Delta\mathbf{w}^\top\left(\frac{\mathbf{V}^\top\boldsymbol{\Sigma}_r\mathbf{U}^\top\boldsymbol{\Pi}\boldsymbol{\Pi}^\top\mathbf{U}\boldsymbol{\Sigma}_r\mathbf{V}^\top}{n}\right)\Delta\mathbf{w} \geq (1-\epsilon_1)\left|\left|\frac{\boldsymbol{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2,$$

and

$$\lambda||\Delta\mathbf{w}||_2^2 \geq \frac{\lambda}{\boldsymbol{\sigma}_{r+1}^2}||\boldsymbol{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}||_2^2 = \frac{\lambda n}{\boldsymbol{\sigma}_{r+1}^2}\left|\left|\frac{\boldsymbol{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2,$$

where

$$\begin{aligned}
2\Delta\mathbf{w}^\top\left(\frac{\mathbf{V}^\top\boldsymbol{\Sigma}_{\bar{r}}\mathbf{U}^\top\boldsymbol{\Pi}\boldsymbol{\Pi}^\top\mathbf{U}\boldsymbol{\Sigma}_r\mathbf{V}^\top}{n}\right)\Delta\mathbf{w} =& 2\Delta\mathbf{w}^\top\left(\frac{\mathbf{V}^\top\boldsymbol{\Sigma}_{\bar{r}}\mathbf{U}_{\bar{r}}^\top\boldsymbol{\Pi}\boldsymbol{\Pi}^\top\mathbf{U}_r\boldsymbol{\Sigma}_r\mathbf{V}^\top}{n}\right)\Delta\mathbf{w} \\
\geq& -\tau_1\left|\left|\frac{\boldsymbol{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2\left|\left|\frac{\boldsymbol{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2,
\end{aligned}$$

we have

$$\begin{aligned}\Delta\mathbf{w}^\top\left(\frac{\mathbf{X}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{X}}{n}+\lambda\mathbf{I}_p\right)\Delta\mathbf{w} &\geq (1-\epsilon_1)\left\|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right\|_2^2+\frac{\lambda n}{\boldsymbol{\sigma}_{r+1}^2}\left\|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right\|_2^2\\ &\quad -2\tau_1\left\|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right\|_2\left\|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right\|_2\\ &\geq\left(\frac{1}{2}-\frac{\epsilon_1}{2}\right)\left\|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right\|_2^2+\frac{\lambda n}{2\sigma_{r+1}^2}\left\|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right\|_2^2.\end{aligned}$$

Consider the term $\left\langle\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\frac{\mathbf{X}\mathbf{w}^*}{\sqrt{n}},-\frac{\mathbf{X}\Delta\mathbf{w}}{\sqrt{n}}\right\rangle$, we have

$$\begin{aligned}\left\langle\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\frac{\mathbf{X}\mathbf{w}^*}{\sqrt{n}},\frac{\mathbf{X}}{\sqrt{n}}(\widehat{\mathbf{w}}_{\text{HS}}-\mathbf{w}^*)\right\rangle &=\left\langle\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\frac{\mathbf{X}_r\mathbf{w}^*}{\sqrt{n}},-\frac{\mathbf{X}_r\Delta\mathbf{w}}{\sqrt{n}}\right\rangle\\ &+\left\langle\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\frac{\mathbf{X}_{\bar{r}}\mathbf{w}^*}{\sqrt{n}},-\frac{\mathbf{X}_r\Delta\mathbf{w}}{\sqrt{n}}\right\rangle\\ &+\left\langle\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\frac{\mathbf{X}_r\mathbf{w}^*}{\sqrt{n}},-\frac{\mathbf{X}_{\bar{r}}\Delta\mathbf{w}}{\sqrt{n}}\right\rangle\\ &+\left\langle\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\frac{\mathbf{X}_{\bar{r}}\mathbf{w}^*}{\sqrt{n}},-\frac{\mathbf{X}_{\bar{r}}\Delta\mathbf{w}}{\sqrt{n}}\right\rangle.\end{aligned}$$

Notice that the random matrix $\mathbf{\Pi}^\top\mathbf{U}_r$ and $\mathbf{\Pi}^\top\mathbf{U}_r$ can be treated as two Gaussian random matrices with entries sampled i.i.d from $\mathcal{N}(0,1/m)$. Applying Lemma 36 and Lemma 37, we can bound above terms separately:

$$\left\langle\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\frac{\mathbf{X}_r\mathbf{w}^*}{\sqrt{n}},-\frac{\mathbf{X}_r\Delta\mathbf{w}}{\sqrt{n}}\right\rangle\leq\epsilon_1\left\|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\mathbf{w}^*}{\sqrt{n}}\right\|_2\left\|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right\|_2,$$

$$\left\langle\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\frac{\mathbf{X}_{\bar{r}}\mathbf{w}^*}{\sqrt{n}},-\frac{\mathbf{X}_r\Delta\mathbf{w}}{\sqrt{n}}\right\rangle\leq\tau_1\left\|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\mathbf{w}^*}{\sqrt{n}}\right\|_2\left\|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right\|_2,$$

$$\left\langle\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\frac{\mathbf{X}_r\mathbf{w}^*}{\sqrt{n}},-\frac{\mathbf{X}_{\bar{r}}\Delta\mathbf{w}}{\sqrt{n}}\right\rangle\leq\tau_1\left\|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\mathbf{w}^*}{\sqrt{n}}\right\|_2\left\|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right\|_2,$$

$$\left\langle\left(\mathbf{\Pi}\mathbf{\Pi}^\top-\mathbf{I}_n\right)\frac{\mathbf{X}_{\bar{r}}\mathbf{w}^*}{\sqrt{n}},-\frac{\mathbf{X}_{\bar{r}}\Delta\mathbf{w}}{\sqrt{n}}\right\rangle\leq\upsilon_1\left\|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\mathbf{w}^*}{\sqrt{n}}\right\|_2\left\|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right\|_2.$$

By Cauchy-Schwarz inequality, we have

$$\begin{aligned}
&\left\langle \left(\mathbf{\Pi}\mathbf{\Pi}^\top - \mathbf{I}_n\right) \frac{\mathbf{X}\mathbf{w}^*}{\sqrt{n}}, \frac{\mathbf{X}}{\sqrt{n}}(\widehat{\mathbf{w}}_{\mathrm{HS}} - \mathbf{w}^*) \right\rangle \\
\leq\ & \epsilon_1 \left\|\frac{\mathbf{\Sigma}_r \mathbf{V}^\top \mathbf{w}^*}{\sqrt{n}}\right\|_2 \left\|\frac{\mathbf{\Sigma}_r \mathbf{V}^\top \Delta\mathbf{w}}{\sqrt{n}}\right\|_2 + \tau_1 \left\|\frac{\mathbf{\Sigma}_{\bar{r}} \mathbf{V}^\top \mathbf{w}^*}{\sqrt{n}}\right\|_2 \left\|\frac{\mathbf{\Sigma}_r \mathbf{V}^\top \Delta\mathbf{w}}{\sqrt{n}}\right\|_2 \\
&+ \tau_1 \left\|\frac{\mathbf{\Sigma}_r \mathbf{V}^\top \mathbf{w}^*}{\sqrt{n}}\right\|_2 \left\|\frac{\mathbf{\Sigma}_{\bar{r}} \mathbf{V}^\top \Delta\mathbf{w}}{\sqrt{n}}\right\|_2 + \upsilon_1 \left\|\frac{\mathbf{\Sigma}_{\bar{r}} \mathbf{V}^\top \mathbf{w}^*}{\sqrt{n}}\right\|_2 \left\|\frac{\mathbf{\Sigma}_{\bar{r}} \mathbf{V}^\top \Delta\mathbf{w}}{\sqrt{n}}\right\|_2 \\
\leq\ & \frac{4\epsilon_1^2}{1-\epsilon_1} \left\|\frac{\mathbf{\Sigma}_r \mathbf{V}^\top \mathbf{w}^*}{\sqrt{n}}\right\|_2^2 + \frac{1-\epsilon_1}{8} \left\|\frac{\mathbf{\Sigma}_r \mathbf{V}^\top \Delta\mathbf{w}}{\sqrt{n}}\right\|_2^2 \\
&+ \frac{4\tau_1^2}{1-\epsilon_1} \left\|\frac{\mathbf{\Sigma}_{\bar{r}} \mathbf{V}^\top \mathbf{w}^*}{\sqrt{n}}\right\|_2^2 + \frac{1-\epsilon_1}{8} \left\|\frac{\mathbf{\Sigma}_r \mathbf{V}^\top \Delta\mathbf{w}}{\sqrt{n}}\right\|_2^2 \\
&+ \frac{4\tau_1^2\sigma_{r+1}^2}{\lambda n} \left\|\frac{\mathbf{\Sigma}_r \mathbf{V}^\top \mathbf{w}^*}{\sqrt{n}}\right\|_2^2 + \frac{\lambda n}{8\sigma_{r+1}^2} \left\|\frac{\mathbf{\Sigma}_{\bar{r}} \mathbf{V}^\top \Delta\mathbf{w}}{\sqrt{n}}\right\|_2^2 \\
&+ \frac{4\upsilon_1^2\sigma_{r+1}^2}{\lambda n} \left\|\frac{\mathbf{\Sigma}_{\bar{r}} \mathbf{V}^\top \mathbf{w}^*}{\sqrt{n}}\right\|_2^2 + \frac{\lambda n}{8\sigma_{r+1}^2} \left\|\frac{\mathbf{\Sigma}_{\bar{r}} \mathbf{V}^\top \Delta\mathbf{w}}{\sqrt{n}}\right\|_2^2.
\end{aligned}$$

From the proof of Theorem 24, we know

$$\begin{aligned}
&\frac{1-\epsilon_1}{2} \left\|\frac{\mathbf{\Sigma}_r \mathbf{V}^\top \Delta\mathbf{w}}{\sqrt{n}}\right\|_2^2 + \frac{\lambda n}{2\sigma_{r+1}^2} \left\|\frac{\mathbf{\Sigma}_{\bar{r}} \mathbf{V}^\top \Delta\mathbf{w}}{\sqrt{n}}\right\|_2^2 \\
\leq\ & \Delta\mathbf{w}^\top \left(\frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{n} + \lambda \mathbf{I}_p \right) \Delta\mathbf{w} \\
=\ & \left\langle \left(\mathbf{\Pi}\mathbf{\Pi}^\top - \mathbf{I}_n\right) \frac{\mathbf{X}\mathbf{w}^*}{\sqrt{n}}, \frac{\mathbf{X}}{\sqrt{n}}(\widehat{\mathbf{w}}_{\mathrm{HS}} - \mathbf{w}^*) \right\rangle .
\end{aligned}$$

Combining the above, we have

$$\begin{aligned}
\frac{1-\epsilon_1}{4}\left|\left|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2 + \frac{\lambda n}{4\sigma_{r+1}^2}\left|\left|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2 \leq& \\
\left(\frac{4\epsilon_1^2}{1-\epsilon_1} + \frac{4\tau_1^2\sigma_{r+1}^2}{\lambda n}\right)\left|\left|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\mathbf{w}^*}{\sqrt{n}}\right|\right|_2^2& \\
+\left(\frac{4\tau_1^2}{1-\epsilon_1} + \frac{4\upsilon_1^2\sigma_{r+1}^2}{\lambda n}\right)\left|\left|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\mathbf{w}^*}{\sqrt{n}}\right|\right|_2^2& \\
\leq \left(\frac{4\epsilon_1^2}{1-\epsilon_1} + \frac{4\tau_1^2\sigma_{r+1}^2}{\lambda n} + \frac{4\tau_1^2\rho^2}{1-\epsilon} + \frac{4\rho^2\upsilon_1^2\sigma_{r+1}^2}{\lambda n}\right)||\mathbf{w}^*||_{\mathbf{X}}^2.&
\end{aligned}$$

Thus

$$\begin{aligned}
||\Delta\mathbf{w}||_{\mathbf{X}}^2 =& \left|\left|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2 + \left|\left|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2 \\
\leq& \left(\frac{4}{1-\epsilon_1} + \frac{4\sigma_{r+1}^2}{\lambda n}\right)\left(\frac{1-\epsilon_1}{4}\left|\left|\frac{\mathbf{\Sigma}_r\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2 + \frac{\lambda n}{4\sigma_{r+1}^2}\left|\left|\frac{\mathbf{\Sigma}_{\bar{r}}\mathbf{V}^\top\Delta\mathbf{w}}{\sqrt{n}}\right|\right|_2^2\right) \\
\leq& \left(\frac{4}{1-\epsilon_1} + \frac{4\sigma_{r+1}^2}{\lambda n}\right)\left(\frac{4\epsilon_1^2}{1-\epsilon_1} + \frac{4\tau_1^2\sigma_{r+1}^2}{\lambda n} + \frac{4\tau_1^2\rho^2}{1-\epsilon_1} + \frac{4\rho^2\upsilon_1^2\sigma_{r+1}^2}{\lambda n}\right)||\mathbf{w}^*||_{\mathbf{X}}^2,
\end{aligned}$$

which concludes the proof.

4.6.3 Proof of Theorem 28

For notation simplicity, let

$$\widetilde{\mathbf{H}} = \frac{\mathbf{X}^\top\mathbf{\Pi}\mathbf{\Pi}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p \quad \text{and} \quad \mathbf{H} = \frac{\mathbf{X}^\top\mathbf{X}}{n} + \lambda\mathbf{I}_p.$$

Based on the property of similarity matrices, we have

$$\kappa(\widetilde{\mathbf{H}}^{-1}\mathbf{H}) = \kappa(\widetilde{\mathbf{H}}^{-1/2}\mathbf{H}\widetilde{\mathbf{H}}^{-1/2}) = \frac{\max_{\mathbf{w}} \mathbf{w}^\top\widetilde{\mathbf{H}}^{-1/2}\mathbf{H}\widetilde{\mathbf{H}}^{-1/2}\mathbf{w}}{\min_{\mathbf{w}} \mathbf{w}^\top\widetilde{\mathbf{H}}^{-1/2}\mathbf{H}\widetilde{\mathbf{H}}^{-1/2}\mathbf{w}}.$$

Consider the quantity $|\mathbf{w}^\top \widetilde{\mathbf{H}}^{-1/2}(\mathbf{H} - \widetilde{\mathbf{H}})\widetilde{\mathbf{H}}^{-1/2}\mathbf{w}|$. We have

$$\begin{aligned}
|\mathbf{w}^\top \widetilde{\mathbf{H}}^{-1/2}(\mathbf{H} - \widetilde{\mathbf{H}})\widetilde{\mathbf{H}}^{-1/2}\mathbf{w}| =& \left\langle \left(\mathbf{H} - \widetilde{\mathbf{H}}\right) \widetilde{\mathbf{H}}^{-1/2}\mathbf{w}, \widetilde{\mathbf{H}}^{-1/2}\mathbf{w} \right\rangle \\
=& \left\langle \left(\mathbf{\Pi}\mathbf{\Pi}^\top - \mathbf{I}_n\right) \frac{\mathbf{X}}{\sqrt{n}}\widetilde{\mathbf{H}}^{-1/2}\mathbf{w}, \frac{\mathbf{X}}{\sqrt{n}}\widetilde{\mathbf{H}}^{-1/2}\mathbf{w} \right\rangle \\
\leq& \rho_2 \left(\mathbf{X}\mathbb{R}^p, \mathbf{\Pi}, \frac{\mathbf{X}}{\sqrt{n}}\widetilde{\mathbf{H}}^{-1/2}\mathbf{w}\right) ||\widetilde{\mathbf{H}}^{-1/2}\mathbf{w}||_{\mathbf{X}}^2 \\
\leq& C_0 \sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m} \log\left(\frac{1}{\delta}\right)} ||\widetilde{\mathbf{H}}^{-1/2}\mathbf{w}||_{\mathbf{X}}^2.
\end{aligned}$$

For any vector $\mathbf{u} \in \mathbb{R}^p$, we have

$$\begin{aligned}
||\widetilde{\mathbf{H}}^{1/2}\mathbf{u}||_2^2 =& \mathbf{u}^\top \left(\frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{n} + \lambda \mathbf{I}_p\right)\mathbf{u} \\
=& \mathbf{u}^\top \left(\frac{\mathbf{X}^\top \mathbf{\Pi}\mathbf{\Pi}^\top \mathbf{X}}{n}\right)\mathbf{u} + \lambda ||\mathbf{u}||_2^2 \\
\geq& \rho_1(\mathbf{X}\mathbb{R}^p, \mathbf{\Pi})||\mathbf{u}||_{\mathbf{X}}^2 \\
\geq& \left(1 - C_0 \sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m} \log\left(\frac{1}{\delta}\right)}\right) ||\mathbf{u}||_{\mathbf{X}}^2.
\end{aligned}$$

Let $\mathbf{u} = \widetilde{\mathbf{H}}^{-1/2}\mathbf{w}$, we have

$$||\widetilde{\mathbf{H}}^{-1/2}\mathbf{w}||_{\mathbf{X}}^2 \leq \frac{1}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m} \log\left(\frac{1}{\delta}\right)}} ||\mathbf{w}||_2^2.$$

Combining, we get

$$|\mathbf{w}^\top \widetilde{\mathbf{H}}^{-1/2}(\mathbf{H} - \widetilde{\mathbf{H}})\widetilde{\mathbf{H}}^{-1/2}\mathbf{w}| \leq \frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m} \log\left(\frac{1}{\delta}\right)}}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m} \log\left(\frac{1}{\delta}\right)}} ||\mathbf{w}||_2^2,$$

which implies

$$\max_{\mathbf{w}} \mathbf{w}^\top \widetilde{\mathbf{H}}^{-1/2}\mathbf{H}\widetilde{\mathbf{H}}^{-1/2}\mathbf{w} \leq ||\mathbf{w}||_2^2 + \frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}||\mathbf{w}||_2^2$$
$$= \frac{1}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}||\mathbf{w}||_2^2,$$

and

$$\min_{\mathbf{w}} \mathbf{w}^\top \widetilde{\mathbf{H}}^{-1/2}\mathbf{H}\widetilde{\mathbf{H}}^{-1/2}\mathbf{w} \geq ||\mathbf{w}||_2^2 - \frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}||\mathbf{w}||_2^2$$
$$= \frac{1 - 2C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}||\mathbf{w}||_2^2.$$

Thus we know

$$\kappa(\widetilde{\mathbf{H}}^{-1}\mathbf{H}) \leq \frac{1}{1 - 2C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p\cap\mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}.$$

The proof for $\kappa_{\mathrm{DRP}}(\mathbf{X}, \mathbf{R}, \lambda)$ is analogous.

4.6.4 Proof of Lemma 30

Note that (4.6) is sketching the following problem

$$\arg\min_{\mathbf{u}} \mathbf{u}^\top \left(\frac{\mathbf{X}^\top\mathbf{X}}{2n} + \frac{\lambda}{2}\mathbf{I}_p\right)\mathbf{u} - \left\langle \frac{\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)})}{n} - \lambda\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)}, \mathbf{u}\right\rangle,$$

where $\mathbf{w}^* - \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)}$ is the optimal solution. Thus applying Theorem 24, We have

$$||\widehat{\mathbf{u}}^{(t)} - (\mathbf{w}^* - \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)})||_{\mathbf{X}} \leq \frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}||\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)} - \mathbf{w}^*||_{\mathbf{X}}.$$

Using the definition that $\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} = \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)} + \widehat{\mathbf{u}}^{(t)}$, we obtain the desired result.

4.6.5 Proof of Theorem 31

By triangle inequality we have the following decomposition:

$$\begin{aligned}
||\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \mathbf{w}^*||_{\mathbf{X}} &\leq ||\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \mathbf{w}^*||_{\mathbf{X}} + ||\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}||_{\mathbf{X}} \\
&\leq \frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}||\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t)} - \mathbf{w}^*||_{\mathbf{X}} + ||\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}||_{\mathbf{X}} \\
&\leq \left(\frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}{1 - C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}\log\left(\frac{1}{\delta}\right)}}\right)^t ||\mathbf{w}^*||_{\mathbf{X}} + ||\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}||_{\mathbf{X}}.
\end{aligned}$$

For the term $||\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}||_{\mathbf{X}}$, we can further bridge $\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}$ and $\widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}$ by $\bar{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}$, which is the result of one exact step of IHS initialized at $\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t)}$. Thus we have the following decomposition

$$||\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}||_{\mathbf{X}} \leq ||\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \bar{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}||_{\mathbf{X}} + ||\bar{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \widehat{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}||_{\mathbf{X}}.$$

Applying the Theorem 24 for DRP we have the following bound for $||\widetilde{\mathbf{w}}_{\text{HS}}^{(t+1)} - \bar{\mathbf{w}}_{\text{HS}}^{(t+1)}||_{\mathbf{X}}$:

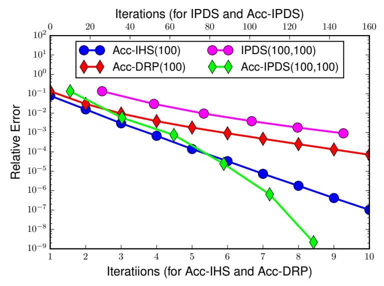
$$
\begin{aligned}
||\widetilde{\mathbf{w}}_{\text{HS}}^{(t+1)} - \bar{\mathbf{w}}_{\text{HS}}^{(t+1)}||_{\mathbf{X}} &\leq \lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)||\widetilde{\mathbf{w}}_{\text{HS}}^{(t+1)} - \bar{\mathbf{w}}_{\text{HS}}^{(t+1)}||_2 \\
&\leq \lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)\left(\frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}{1-C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}\right)^k ||\bar{\mathbf{w}}_{\text{HS}}^{(t+1)}||_2 \\
&\leq \lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)\left(\frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}{1-C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}\right)^k \left(||\bar{\mathbf{w}}_{\text{HS}}^{(t+1)} - \mathbf{w}^*||_2 + ||\mathbf{w}^*||_2\right) \\
&\leq 2\lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)\left(\frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}{1-C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}\right)^k ||\mathbf{w}^*||_2.
\end{aligned}
$$

We can relate the error $||\bar{\mathbf{w}}_{\text{HS}}^{(t+1)} - \widehat{\mathbf{w}}_{\text{HS}}^{(t+1)}||_{\mathbf{X}}$ to the error term at t-th outer loop iteration: $||\widetilde{\mathbf{w}}_{\text{HS}}^{(t)} - \widehat{\mathbf{w}}_{\text{HS}}^{(t)}||_{\mathbf{X}}$:

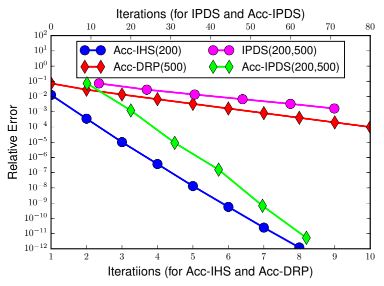
$$
\begin{aligned}
||\bar{\mathbf{w}}_{\text{HS}}^{(t+1)} - \widehat{\mathbf{w}}_{\text{HS}}^{(t+1)}||_{\mathbf{X}} =&||\widetilde{\mathbf{w}}_{\text{HS}}^{(t)} - \widetilde{\mathbf{H}}^{-1}\nabla P(\widetilde{\mathbf{w}}_{\text{HS}}^{(t)}) - \widehat{\mathbf{w}}_{\text{HS}}^{(t)} - \widetilde{\mathbf{H}}^{-1}\nabla P(\widehat{\mathbf{w}}_{\text{HS}}^{(t)})||_{\mathbf{X}} \\
=&||\widetilde{\mathbf{H}}^{-1}(\widetilde{\mathbf{H}} - \mathbf{H})(\widetilde{\mathbf{w}}_{\text{HS}}^{(t)} - \widehat{\mathbf{w}}_{\text{HS}}^{(t)})||_{\mathbf{X}} \\
\leq&||\widetilde{\mathbf{H}}^{-1}||_2||\widetilde{\mathbf{H}} - \mathbf{H}||_2||\widetilde{\mathbf{w}}_{\text{HS}}^{(t)} - \widehat{\mathbf{w}}_{\text{HS}}^{(t)}||_{\mathbf{X}} \\
\leq&\frac{4\lambda_{\max}\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)}{\lambda}||\widetilde{\mathbf{w}}_{\text{HS}}^{(t)} - \widehat{\mathbf{w}}_{\text{HS}}^{(t)}||_{\mathbf{X}} \\
\leq&\frac{8\lambda_{\max}^2\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)}{\lambda}\left(\frac{C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}{1-C_0\sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n\cap\mathcal{S}^{p-1})}{d}}\log\left(\frac{1}{\delta}\right)}\right)^k ||\mathbf{w}^*||_2.
\end{aligned}
$$

Combining above inequalities we obtained the following iterative error bound for $\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)}$:

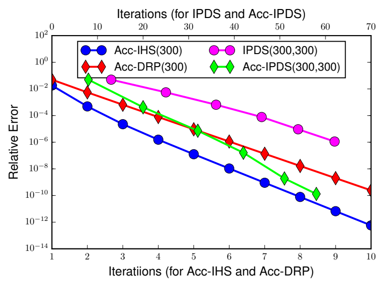
$$
\begin{aligned}
||\widetilde{\mathbf{w}}_{\mathrm{HS}}^{(t+1)} - \mathbf{w}^*||_{\mathbf{X}} \leq & \left(\frac{C_0 \sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}} \log\left(\frac{1}{\delta}\right)}{1 - C_0 \sqrt{\frac{\mathbb{W}^2(\mathbf{X}\mathbb{R}^p \cap \mathcal{S}^{n-1})}{m}} \log\left(\frac{1}{\delta}\right)} \right)^t ||\mathbf{w}^*||_{\mathbf{X}} \\
& + \frac{10\lambda_{\max}^2\left(\frac{\mathbf{X}^\top\mathbf{X}}{n}\right)}{\lambda} \left(\frac{C_0 \sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1})}{d}} \log\left(\frac{1}{\delta}\right)}{1 - C_0 \sqrt{\frac{\mathbb{W}^2(\mathbf{X}^\top\mathbb{R}^n \cap \mathcal{S}^{p-1})}{d}} \log\left(\frac{1}{\delta}\right)} \right)^k ||\mathbf{w}^*||_2.
\end{aligned}
$$



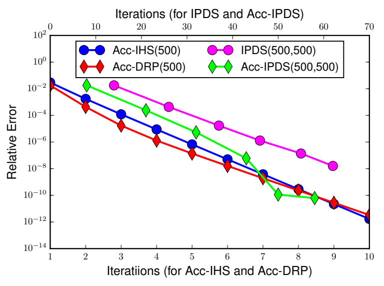
$(n, p) = (1000, 1000)$



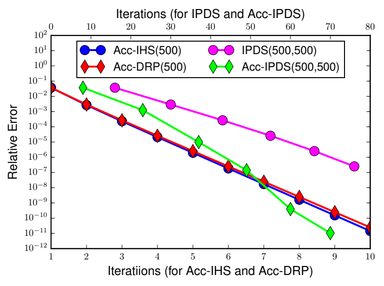
$(n, p) = (2000, 10000)$



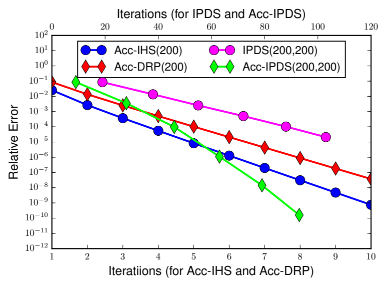
$(n, p) = (5000, 5000)$



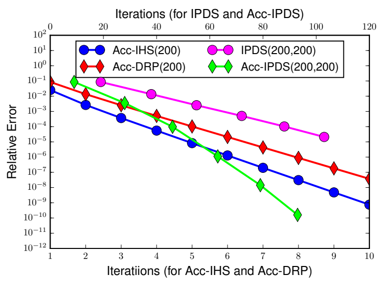
$(n, p) = (5000, 10000)$



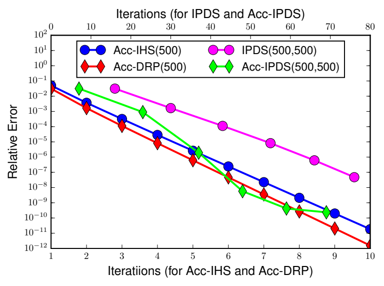
$(n, p) = (5000, 20000)$



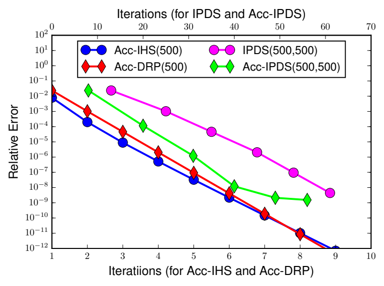
$(n, p) = (10000, 1000)$



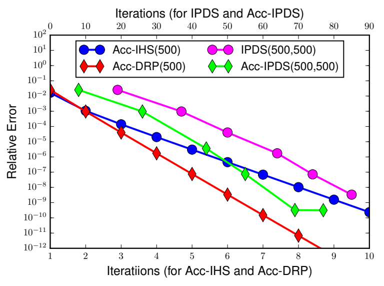
$(n, p) = (10000, 1000)$



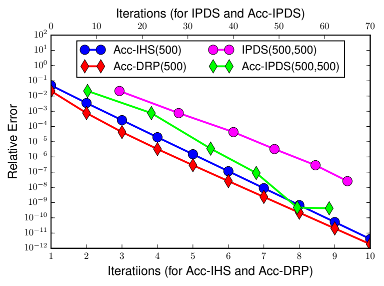
$(n, p) = (10000, 20000)$



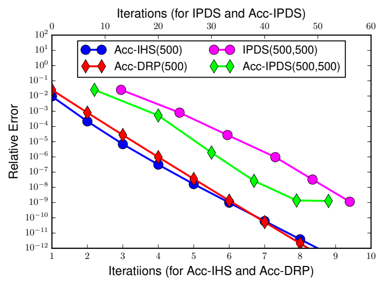
$(n, p) = (20000, 2000)$



$(n, p) = (20000, 5000)$

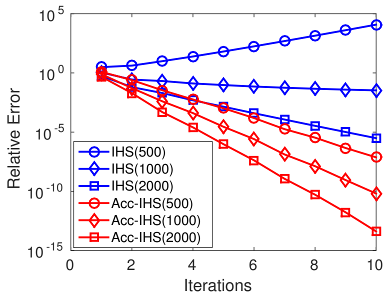


$(n, p) = (20000, 10000)$

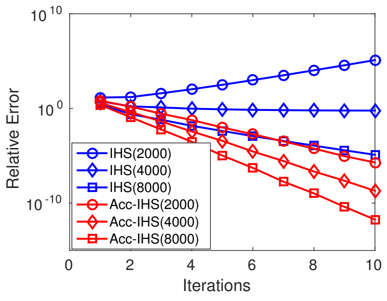


$(n, p) = (20000, 20000)$

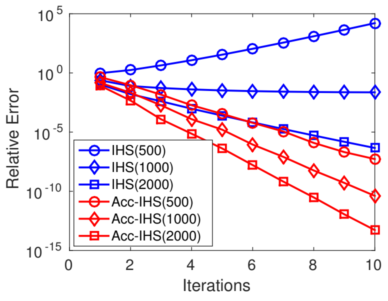
Figure 4.3: Comparion of IPDS and Acc-IPDS versus with IHS and DRP various simulated datasets.



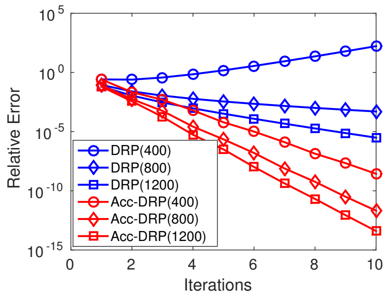
connect4



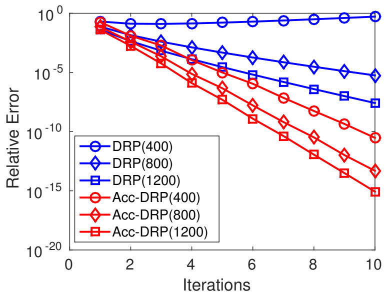
slice



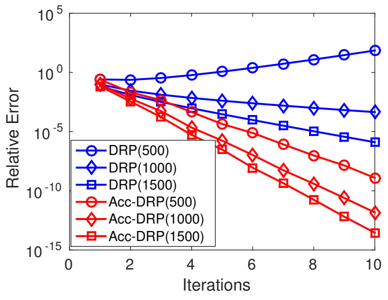
year



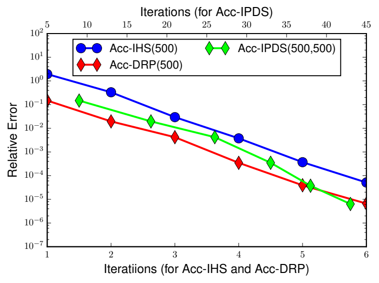
colon-cancer



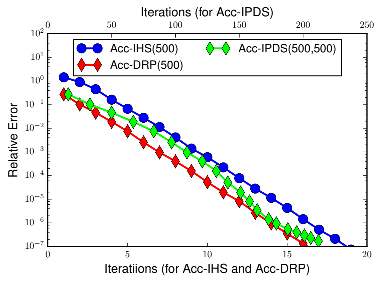
duke breast-cancer



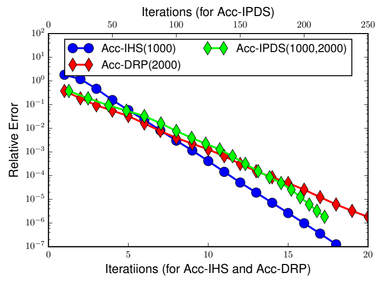
leukemia



cifar

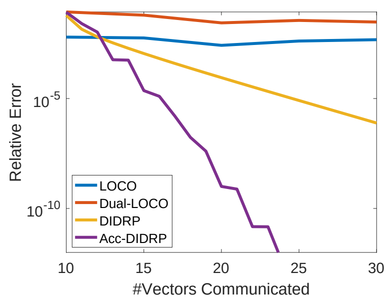


gisette

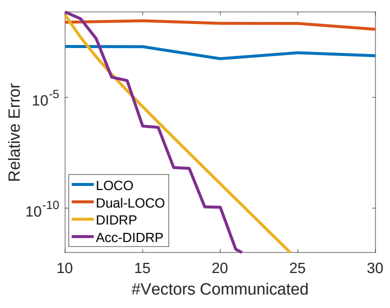


sector

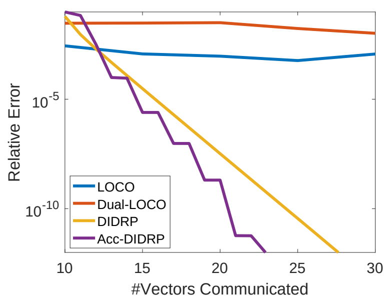
Figure 4.4: Comparion of various iterative sketching approaches on real-world data sets. Top row: Acc-IHS versus IHS, middle row: Acc-DRP versus DRP, bottom row: Acc-IPDS versus Acc-IHS and Acc-DRP.



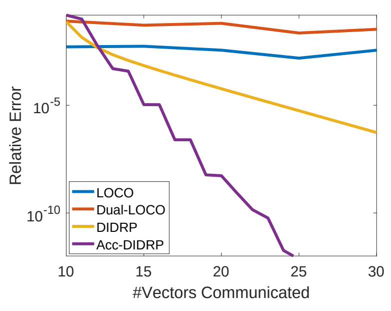
(1000,1000,10)



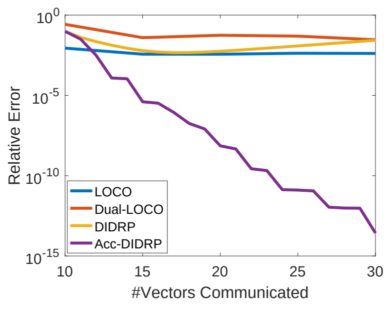
(2000,500,20)



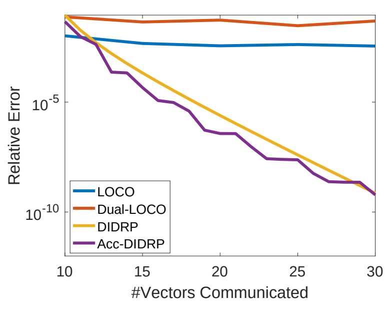
(2000,1000,20)



(2000,2000,120)

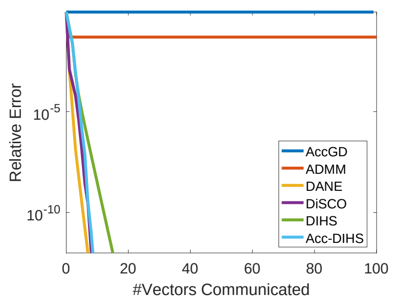


(3000,3000,10)

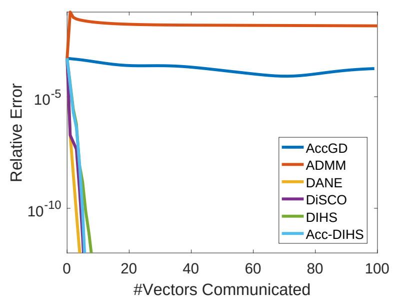


(5000,1000,20)

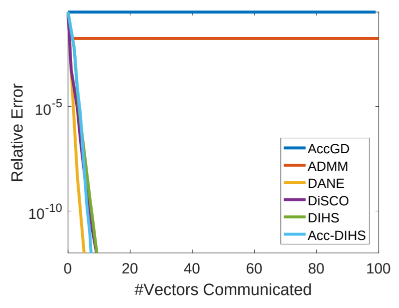
Figure 4.5: Comparion of various approaches for distributed optimization under the partition by feature scenario, with different settings of (n, p, m).



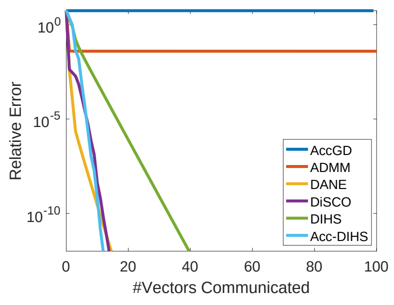
(10000,100,20)



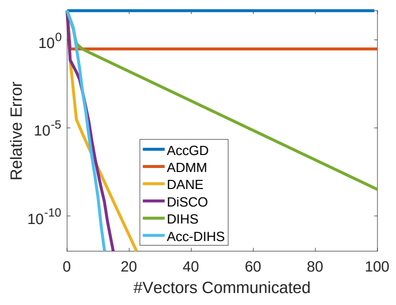
(20000,20,20)



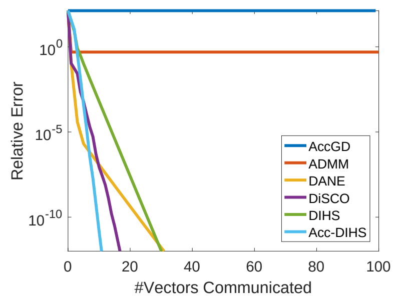
(20000,50,20)



(20000,50,200)

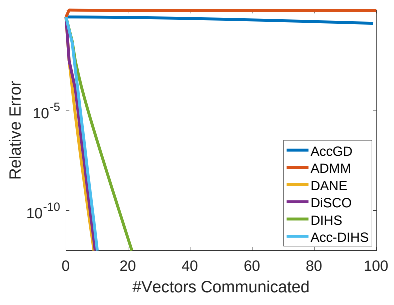


20000,500,200)

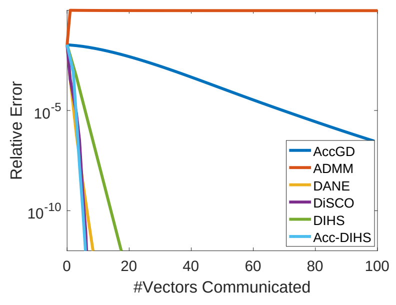


(50000,500,500)

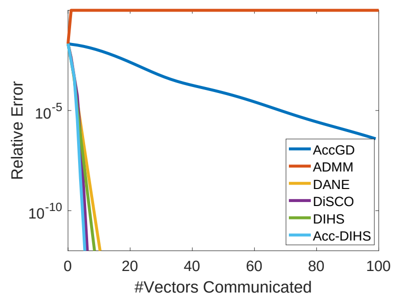
Figure 4.6: Comparion of various approaches for distributed optimization under the partition by sample scenario, with different settings of (n, p, m) (cont.).



mnist



toms



twitter

Figure 4.7: Comparion of various approaches for distributed optimization on several real world data sets under the partition by sample setting.

CHAPTER 5
COMMUNICATION-COMPUTATION BALANCED OPTIMIZATION

5.1 Motivation

Distributed computing systems are widely used to train machine learning models due to large data sets collected across sciences. When a data set cannot fit into the memory of a single machine or the learning process is too time consuming, the data set is distributed across multiple computation nodes and a distributed optimization algorithm is used to speedup the training process. When developing a distributed optimization algorithm, in order to keep the runtime low, one needs to consider two important factors: the communication and computation costs. The communication cost arises from multiple machines exchanging their local information to reach a global consensus, while the computation cost accounts for each machine processing the local data. A typical optimization algorithm consists of a number of iterative steps that might require multiple rounds of communication and within each round of communication a number of local computations are performed. When designing an efficient distributed optimization method there is a fundamental challenge in balancing efficiency in communication and computation, as they often conflict with each other.

In this chapter, we consider the problem of distributed optimization of finite-sums, which appears ubiquitously in machine learning [Bottou et al., 2016]. To the best of our knowledge, even if one just considers the computational costs, while ignoring communication, no existing distributed optimization algorithm can obtain linear speedup in computation compared to any single machine algorithm when the condition number is large. Concretely, we consider a setting with m-machines, each holding n individual functions, and the goal is to minimize the following global average:

$$\min_{w \in \mathbb{R}^d} f(w) := \frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} f_{ij}(w) = \frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} \ell(w^\top a_{ij}, b_{ij}) + g(w), \tag{5.1}$$

where $f_{ij}(w)$ is the i-th individual function on j-th machine. This problem arises in, for example, regularized empirical risk minimization. Let $\{a_{ij}, b_{ij}\}_{i \in [n], j \in [m]}$ be i.i.d. samples from an unknown distribution $\mathcal{D}$, where a_{ij} represents observed feature vectors and b_{ij} represents responses of interest. By setting $f_{ij}(w) = \ell(w^\top a_{ij}, b_{ij}) + \lambda g(w)$, where $\ell(w^\top a, b)$ is a loss function and $g(w)$ is a regularization term, the optimization problem (5.1) can be used for fitting generalized linear models used in classification and regression. For example, when $\ell(w^\top a, b) = \log(1 + \exp(-(w^\top a)b))$ and $g(w) = (1/2)||w||^2$, one gets ℓ_2 regularized logistic regression, while when $\ell(w^\top a, b) = (1/2)(w^\top a - b)^2$, and $g(w) = (1/2)||w||_1$ one obtains the lasso objective.

Our contributions. We propose novel distributed optimization algorithms to minimize the convex finite-sum objective in (5.1). The approach provably achieves near-linear speedup in computation, even when the condition number is large. Compared with previous attempts, our proposed approaches obtain a communication-computation balance that is beneficial for improving the overall runtime efficiency.

Notation. For a vector $w \in \mathbb{R}^d$, we use $||w||$ to denote its ℓ_2 norm and $||w||_1$ to denote its ℓ_1 norm. For a matrix A, we use trace(A) to denote the trace of A. For two sequences of numbers $\{a_n\}$ and $\{b_n\}$, we say $a_n = \mathcal{O}(b_n)$ if $a_n \leq C b_n$ for n large enough, with some positive constant C, and use the notation $\widetilde{\mathcal{O}}(\cdot)$ to hide polylogarithmic factors. We say $a_n = \Omega(b_n)$ if $b_n = \mathcal{O}(a_n)$. We denote $a_n \asymp b_n$ if $a_n = \mathcal{O}(b_n)$ and $b_n = \mathcal{O}(a_n)$. Finally, we use I to denote an identity matrix, and use $[n]$ to denote the set $\{1, ..., n\}$.

5.2 Problem Set-up and related work

Throughout this chapter, we focus on objectives $f(w)$ that satisfy the following smoothness and strong convexity conditions.

Assumption 38. *The function $f(w)$ admits the finite-sum structure $f(w) = (1/mn) \sum_{j=1}^{m} \sum_{i=1}^{n} \ell(w^\top a_{ij}, b_{ij}) + g(w)$, where the loss function $\ell(w^\top a, b)$ is convex and L-smooth, the regularization function $g(w)$ is convex, and the overall objective $f(w)$ is λ-strongly convex.*

- *Every loss function $\ell(w^\top a_{ij}, b_{ij})$ is differentiable and its gradient is L-Lipschitz continuous with respect to w: $\forall i \in [n], j \in [m], w_1, w_2 \in \mathbb{R}^d$, we have*

$$|\nabla \ell(w_1^\top a_{ij}, b_{ij}) - \nabla \ell(w_2^\top a_{ij}, b_{ij})| \leq L||w_1 - w_2||.$$

- *The overall objective $f(w)$ is λ-strongly convex with respect to w: $\forall w_1, w_2 \in \mathbb{R}^d$, we have*

$$f(w_1) \geq f(w_2) + \langle \nabla f(w_2), w_2 - w_1 \rangle + \frac{\lambda}{2}||w_1 - w_2||_2^2.$$

Note that for non-strongly convex or non-smooth objectives, we can add a small quadratic term and approximate the original problem via a strongly convex objective, or we can transform the non-smooth loss function to a smooth one using the smoothing technique of Allen-Zhu and Hazan [2016]. The corresponding guarantees can be established as below then.

Recall that there are m machines and the j-th machine has access to $\{f_{ij}(w)\}_{i=1}^{n}$. We consider the synchronous message passing model for distributed computing [Gropp et al., 1996, Dean and Ghemawat, 2008]. In this model, in every round of communication each machine can broadcast (send to all others) information linear in the problem's dimension d. An example of such communication would be the case where each machine communicates its local solution or a gradient vector. Between two communication rounds, each machine performs computations based only on their local information and previously received messages. To quantify the overall runtime of a distributed optimization algorithm, we use the following quantity:

$$\text{Time} = \text{Communication rounds} \times (\#\text{Communication} + \#\text{Computation}),$$

where #Communication and #Computation represents per round time for communication and time for local computation, respectively. In particular, we can measure #Communication by the number of vectors in $\mathbb{R}^d$ transmitted in one communication round, and measure #Computation by the number of parallel gradient calculations performed between two rounds, since gradient calculations are the main computational part in any first-order optimization algorithms.

A number of approaches have been proposed recently to solve (5.1) in a distributed setting under Assumption 38. Ma et al. [2015], Smith et al. [2016], and Zheng et al. [2017] considered the dual formulation of (5.1) and proposed randomized distributed dual coordinate methods (for example, `CoCoA`, `CoCoA`$^+$, `DADM`). Alternating direction method of multipliers (`ADMM`) [Boyd et al., 2011] can be used to naturally formulate the distributed optimization problem in an augmented Lagrangian form, but the best known efficiency guarantees do not improve over accelerated gradient methods [Deng and Yin, 2016, Makhdoumi and Ozdaglar, 2017]. Shamir et al. [2014], Zhang and Xiao [2015], Reddi et al. [2016], and Wang et al. [2017e] focused on provable communication efficient approaches that explored similarity between local objectives and the fact that all the data are i.i.d. from an unknown, fixed distribution (for example, `DANE`, `DiSCO`, `AIDE`, `GIANT`). These methods provably improve communication efficiency over accelerated gradient methods [Nesterov, 1983], but are computationally not as efficient as stochastic optimization methods as they require each machine to process the whole local data between two communication rounds. To achieve both communication and computation efficiency, Lee et al. [2017a] proposed the distributed stochastic variance reduced gradient method (DSVRG), which extends the SVRG method [Johnson and Zhang, 2013] in a distributed fashion. DSVRG uses all machines to compute in parallel the "reference full gradient", while one machine performs stochastic gradient updates, using the "reference full gradient" to reduce the variance of updates. However, this method is not practical as it requires extra data to be stored on each machine in order to simulate uniform sampling with replacement in SVRG. Shamir [2016] fixed this issue by proving that sampling without replacement can be used in SVRG to achieve the same rate of convergence. Unfortunately, the convergence result only holds for quadratic objectives. In summary, existing DSVRG methods have the following communication and computation efficiency guarantees to solve (5.1).

Theorem 39 (Lee et al. 2017a, Shamir 2016)**.** *Suppose the objective in* (5.1) *is quadratic and satisfies Assumption 38. DSVRG outputs a solution $\widehat{w}$ satisfying $f(\widehat{w}) - f(w^*) \leq \epsilon$ in $\mathcal{O}\left((1 + L/(\lambda n)) \log(1/\epsilon)\right)$ rounds of communication, with each machine performing $\mathcal{O}\left((n + L/\lambda) \log(1/\epsilon)\right)$ parallel operations.*

When the condition number is small, $L/\lambda \leq n$, Theorem 39 states that DSVRG only requires $\widetilde{\mathcal{O}}(1)$ rounds of communication and $\widetilde{\mathcal{O}}(n)$ parallel computations. Ignoring log-terms, this rate is optimal in terms of both communication and computation, since simply making one pass over the whole data sets would require $\mathcal{O}(n)$ parallel computations. However, when the condition number is large, the above rate can be bad in terms of both communication and computation. Lee et al. [2017a] proposed to fix this issue using an accelerated approach, called DASVRG, which combines the DSVRG algorithm with an accelerated proximal point technique [Shalev-Shwartz and Zhang, 2016, Lin et al., 2015, Frostig et al., 2015]. When the condition number is big, $L/\lambda \geq n$, DASVRG improves over DSVRG as it needs $\mathcal{O}((1 + \sqrt{L/(\lambda n)}) \log^2(1/\epsilon))$ rounds of communication and $\mathcal{O}((n + \sqrt{(nL)/\lambda}) \log^2(1/\epsilon))$ parallel computations. Lee et al. [2017a] showed that $\Omega(\sqrt{L/(\lambda n)})$ rounds of communication are necessary for a wide range of first-order algorithms, even when local second-order information is used for preconditioning. Therefore, DASVRG is optimal in terms of rounds of communication in the "large condition number" regime, but it does not obtain linear speed-up in computation as we discuss next. First, for single machine optimization problems both upper bound and lower bound are well understood, the cost of solving (5.1) on a single machine using accelerated SVRG is $\mathcal{O}((mn + \sqrt{(mnL)/\lambda}) \log^2(1/\epsilon))$ [Lin et al., 2015]. This computational complexity is almost tight for any randomized optimization method that uses gradient and proximal oracles, as a lower bound is $\Omega(mn + \sqrt{(mnL)/\lambda} \log(1/\epsilon))$ [Woodworth and Srebro, 2016]. For multiple machines setting, we have the following corollary stating a lower bound on the computational cost for distributed optimization of finite-sums, which is a consequence of the single machine computational lower bound.

Corollary 40. ***(Computational lower bound for distributed optimization of finite sums)*** *Suppose the objective to minimize is in form* (5.1) *and there are m machines, each machine holding n unique functions. Then for any algorithm with only access to first order oracle of f_{ij}, it must make at least $\Omega(n + \sqrt{(nL)/(m\lambda)} \log(1/\epsilon))$ parallel gradient computations in order to achieve ϵ objective suboptimality for any objective* (5.1) *that satisfies Assumption 38.*

By comparing the computation cost of running accelerated SVRG on a single machine with that of running DASVRG on m parallel machines, we observe that the computation cost is only reduced by a $\sqrt{m}$ factor — from $\widetilde{\mathcal{O}}(\sqrt{(mnL)/\lambda})$ to $\widetilde{\mathcal{O}}(\sqrt{(nL)/\lambda})$, thus the computation efficiency of DASVRG does not match the lower bound. Moreover, between two communication rounds DSVRG/DASVRG only use one machine to perform stochastic gradient update on its local data, while other machines just wait and do not perform any computation, leading to a waste of computing resources. Another issue with DSVRG/DASVRG is imbalance between communication and computation cost. Though communicating a vector is much more costly than vector floating-point operations, the number of vector floating-point operations is a larger by a factor of n compared to the number of vector communications in DSVRG/DASVRG algorithms. In other words, DSVRG/DASVRG uses machine to make one pass computation on whole local data between two communication rounds[1]. Thus the overall computation is still the main bottleneck in the overall efficiency in a typical distributed system. Our work obtains a better communication-computation balance in a distributed system. Recent work of Tsianos et al. [2012] and Berahas et al. [2017] also consider achieving communication-computation balance in a distributed optimization without exploiting the special finite-sum structure that arises in machine learning problems. By exploiting this structure, our proposed algorithms and corresponding guarantees are substantially different.

5.3 Proposed algorithms

Our first approach to achieve better computation efficiency is through minibatching. The algorithm is similar to the DSVRG method, which uses parallel computing to calculate the "reference full gradient", but differs from DSVRG in how the stochastic variance-reduced updates are performed. Instead of using one machine to perform sequential stochastic gradient update, while other machines wait, we use all machines to calculate the variance-reduced stochastic gradient in parallel, and communicate local averages of the gradient vectors to the master machine. Through this averaging scheme, the variance is further reduced, thus it further improves the convergence over serial SVRG.

Details of the algorithm, named AMD-SVRG, are given in Algorithm 13. The algorithm is divided into R stages and in each stage we approximately solve a proximal point problem:

$$\min_{w} f(w) + \frac{\alpha}{2}||w - y_{r-1}||^2. \tag{5.2}$$

Here y_{r-1} is a vector that linearly depends on previous iterates and α is a tuning parameter. The proximal point problem (5.2) is easier to solve for larger α values as the quadratic penalty

1. It is also the case for other communication efficient approaches, such as DANE, DiSCO, AIDE.

Algorithm 13 AMD-SVRG method.

```
Initialize $w_0, y_0$, $\nu_0 = \sqrt{\lambda/(\lambda+\alpha)}$.
for $r = 1, 2, \ldots, R$ do
  $\widetilde{x}_0 \leftarrow y_{r-1}$.
  for $k = 1, 2, \ldots, K$ do
    All machines perform one round of communication to compute the average gradient $\widetilde{v} \leftarrow (1/mn)\sum_{j=1}^{m}\sum_{i=1}^{n}\nabla\ell_{ij}(\widetilde{x}_{k-1})$.
    Initialize $x_0, z_0 \leftarrow \widetilde{x}_{k-1}$, $\beta_t = (1-\sqrt{(\lambda+\alpha)\eta})/(1+\sqrt{(\lambda+\alpha)\eta})$.
    for $t = 1, 2, \ldots, T$ do
      1. Each machine $j$ draws a minibatch $I_t^{(j)}$ of $b$ samples from the local data set, compute and communicate $\frac{1}{b}\sum_{i\in I_t^{(j)}}\nabla\ell_{ij}(z_{t-1})$.
      2. Master compute
      $v_{t-1} \leftarrow \frac{1}{mb}\sum_{j=1}^{m}\sum_{i\in I_t^{(j)}}\nabla\ell_{ij}(z_{t-1}) + \widetilde{v} - \frac{1}{mb}\sum_{j=1}^{m}\sum_{i\in I_t^{(j)}}\nabla\ell_{ij}(\widetilde{x}_{k-1})$.
      3. Master compute
      $x_t \leftarrow \mathrm{argmin}_x\langle v_{t-1}, x\rangle + \frac{1}{2\eta}||x-z_{t-1}||^2 + \frac{\alpha}{2}||x-y_{r-1}||^2 + g(x)$.
      4. Master update $z_t = x_t + \beta_t(x_t - x_{t-1})$, and broadcast $z_t$.
    end for
    Update $\widetilde{x}_k \leftarrow x_T$.
  end for
  Update $w_r \leftarrow \widetilde{x}_K$.
  Compute $\nu_r \in (0,1)$ such that $\nu_r^2 = (1-\nu_r)\nu_{r-1}^2 + \lambda\nu_k/(\lambda+\alpha)$, and compute $y_r = w_r + \left(\frac{\nu_{r-1}(1-\nu_{r-1})}{\nu_r+\nu_{r-1}^2}\right)(w_r - w_{r-1})$.
end for
Output: $w_R$.
```

makes the strong convexity parameter larger. However, by setting a smaller α value, one needs fewer proximal point steps to approximate the minimizer of $f(w)$. To efficiently find an approximate minimizer of (5.2), we use parallel computing to calculate a minibatch gradient direction and, to save on communication cost, an acceleration scheme accompanied with a large minibatch size. The acceleration technique was used proposed in single machine setting [Nitanda, 2014]. We have the following communication and computation cost guarantees for our double acceleration technique in Algorithm 13.

Theorem 41. *Suppose the objective in* (5.1) *is quadratic and satisfies Assumption 38. The* ***AMD-SVRG*** *method (Algorithm 13) with parameters* α *and* b*, the stepsize* $\eta \asymp \left\{1/L, (b^2(\lambda+\alpha))/L^2\right\}$

with

$$\widetilde{\mathcal{O}}\left(\sqrt{\frac{\lambda+\alpha}{\lambda}}\cdot\left(1+\max\left\{\frac{L}{(\lambda+\alpha)mb},\sqrt{\frac{L}{\lambda+\alpha}}\right\}\right)\right)$$

rounds of communication, and

$$\widetilde{\mathcal{O}}\left(\sqrt{\frac{\lambda+\alpha}{\lambda}}\cdot\left(n+\max\left\{\frac{L}{(\lambda+\alpha)m},b\sqrt{\frac{L}{\lambda+\alpha}}\right\}\right)\right)$$

parallel computation will find a solution $\widehat{w}$ such that $f(\widehat{w})-f(w^*)\leq\epsilon$.

An immediate corollary of Theorem 41 is that the `AMD-SVRG` method always obtains near linear speedup in computation compared with the single machine solution, irrespective of how large the condition number is, when α and b are chosen appropriately.

Corollary 42. *Suppose the condition of Theorem 41 are satisfied. In addition, suppose that $\alpha \asymp \frac{L}{mn}$ and $b \asymp \sqrt{\frac{n}{m}}$. Then the* ***`AMD-SVRG`*** *method with*

$$\widetilde{\mathcal{O}}\left(\left(\sqrt{mn}+\sqrt{\frac{L}{\lambda}}\right)\right)$$

rounds of communication, and

$$\widetilde{\mathcal{O}}\left(\left(n+\sqrt{\frac{Ln}{\lambda m}}\right)\right)$$

parallel computation will find a solution $\widehat{w}$ such that $f(\widehat{w})-f(w^*)\leq\epsilon$.

Corollary 42 establishes that the parallel computational cost for Algorithm 13 is reduced by a factor of m (ignoring the log factors) compared with the best single machine algorithm (accelerated `SVRG`), which is $\mathcal{O}((mn+\sqrt{(mnL)/L})\log(1/\epsilon))$, it matches the computational lower bound of Corollary 40 thus is optimal up to logarithmic factors. Moreover, the ratio between computation/communication operations in `AMD-SVRG` is $\mathcal{O}(\sqrt{n/m})$, which is more balanced compared with `DSVRG`/`DASVRG`.

Extensions to non-strongly convex or non-smooth objectives. Our results can be readily extended to objectives that are non-strongly convex. The main technique to handle this is setting is by adding a small strongly-convex regularization, such as the squared ℓ_2 norm. For distributed finite-sum problems, we can also the `AdaptReg` technique [Allen-Zhu and Hazan, 2016] to obtain the following guarantees, which is also optimal in terms of computation.

Corollary 43. *Suppose the `AMD-SVRG` methods is used together with `AdaptReg` technique on a distributed finite-sum optimization problem* (5.1) *with a quadratic objectives. If $\ell(w^\top a, b)$ is L-smooth and $||w_0 - w^*|| \leq B$, where w_0 is the initialization point, then $\widetilde{\mathcal{O}}(\sqrt{mn} + \sqrt{L/\epsilon}B)$ rounds of communication and $\widetilde{\mathcal{O}}(n + \sqrt{nL/(m\epsilon)}B)$ parallel computation is sufficient to obtain ϵ-objective suboptimality.*

Connection to a practical version of `DSVRG`. Konečnỳ et al. [2016] proposed a practical version of `DSVRG` (Federated `SVRG` or `FSVRG`) to stop wasting computation resources during the stochastic update stage in `DSVRG` method. `FSVRG` uses all machines to run stochastic variance reduced gradient updates and then averages the iterates from all machines to obtain the iterate for the next stage. Konečnỳ et al. [2016] showed that `FSVRG` worked well in practice, but did not analyzed its theoretical convergence. `AMD-SVRG` is similar to `FSVRG` in that both use all machines to perform the stochastic updates, but `FSVRG` sequentially updates the iterate, while `AMD-SVRG` computes the minibatch gradient to obtain one-step update. Furthermore, `AMD-SVRG` uses the momentum techniques to obtain faster convergence. Combining the idea behind `FSVRG` with `AMD-SVRG`, we can obtain a more practical version of `AMD-SVRG` (we name it `AMD-SVRG-P`, see Appendix). The main idea is to run sequential stochastic updates on a local machine instead of taking one step minibatch average gradient. By using the fresher gradient information, we are able to empirically demonstrate improved efficiency in communication. See Section 7.5 for details.

From Corollary 42 we know that Algorithm 13 always achieves near linear computational

Table 5.1: Comparison of communication and computation cost of various distributed optimization approaches for solving finite-sum optimization problem (5.1) with quadratic objectives. It is assumed that $L/\lambda \gg mn$, $n > m$, and the comparison ignores log factors.

Algorithm	Communication	Computation
Ideal (lower bound)	$\sqrt{L/(n\lambda)}$	$\sqrt{(nL)/(m\lambda)}$
`CoCoA+` [Ma et al., 2015, Smith et al., 2016]	L/λ	L/λ
`DANE` [Shamir et al., 2014]	$L^2/(\lambda^2 n)$	L^2/λ^2
`AccGD` [Nesterov, 1983]/`ADMM` [Boyd et al., 2011]	$\sqrt{L/\lambda}$	$n\sqrt{L/\lambda}$
`DiSCO` [Zhang and Xiao, 2015]/`AIDE` [Reddi et al., 2016]	$\sqrt{L/(\sqrt{n}\lambda)}$	$n^{3/4}\sqrt{L/\lambda}$
`DSVRG` [Lee et al., 2017a]	$L/(n\lambda)$	L/λ
`DASVRG` [Lee et al., 2017a]	$\sqrt{L/(n\lambda)}$	$\sqrt{(nL)/\lambda}$
`AMD-SVRG`	$\sqrt{L/\lambda}$	$\sqrt{(nL)/(m\lambda)}$
`AMD-SVRP`	$\sqrt{\min\left\{1, \sqrt{(\rho_{\tilde{\lambda}}^2 d_{\tilde{\lambda}} m)/n}\right\} (L/\lambda)}$	$\sqrt{(nL)/(m\lambda)}$

speedup, regardless of how large the condition number is. However, the communication cost is considerably higher than that of `DASVRG` (by a factor of $\sqrt{n}$). In the next section, we explore better approaches to further reduce the communication cost, while maintaining near linear speedup in terms of computation.

5.3.1 *The accelerated minibatch distributed proximal method*

In this section, we propose to use proximal methods as a precondition step to improve the communication efficiency of `AMD-SVRG` method. Recall that `AMD-SVRG` method uses minibatching technique to achieves optimal computational efficiency, while we can further reduce the required communication rounds by incorporating the second-order information in the minibatch. Details of the algorithm `AMD-SVRP` are given in Algorithm 15 in Appendix. The main idea is motivated by Wang and Zhang [2017], who incorporate subsampled second-order and variance-reduced first order information through a minibatch proximal step. At every iteration, instead of just performing a minibatch variance-reduced gradient step, we solve a proximal problem that incorporates second-order information. With this preconditioning

Algorithm 14 `AMD-SVRG-P`: A practical variant of Accelerated Minibatch Distributed `SVRG` method.

```
Initialize $w_0, y_0$, $\nu_0 = \sqrt{\lambda/(\lambda+\alpha)}$.
for $r = 1, 2, \ldots, R$ do
  $\widetilde{x}_0 \leftarrow y_{r-1}$.
  for $k = 1, 2, \ldots, K$ do
    All machines perform one round of communication to compute the average gradient
        $$\widetilde{v} \leftarrow \frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n} \nabla \ell_{ij}(\widetilde{x}_{k-1}).$$
    $x_0, z_0 \leftarrow \widetilde{x}_{k-1}$, $\beta_t = \frac{1-\sqrt{(\lambda+\alpha)\eta}}{1+\sqrt{(\lambda+\alpha)\eta}}$.
    for $t = 1, 2, \ldots, T$ do
      1. Each machine $j$ draws a minibatch $I_t^{(j)}$ of $b$ samples from the local data set.
      2. Each machine initialize $x_t^{(j)} \leftarrow z_{t-1}$, and perform the following sequential update
      in parallel: for $i \in I_t^{(j)}$ do
        $$x_t^{(j)} \leftarrow \operatorname*{argmin}_x \langle \nabla \ell_{ij}(x_t^{(j)}) + \widetilde{v} - \nabla \ell_{ij}(\widetilde{x}_{k-1}), x\rangle + \frac{1}{2\eta}||x - x_t^{(j)}||^2 + \frac{\alpha}{2}||x - y_{r-1}||^2 + g(x).$$
      3. Each machine send $x_t^{(j)}$ to master.
      4. Master compute
        $$x_t \leftarrow \frac{1}{m}\sum_{j=1}^{m} x_t^{(j)}.$$
      5. Master update
        $$z_t = x_t + \beta_t(x_t - x_{t-1}),$$
      and broadcast $z_t$.
    end for
    Update $\widetilde{x}_k \leftarrow x_T$.
  end for
  Update $w_r \leftarrow \widetilde{x}_K$.
  Compute $\nu_r \in (0,1)$ such that $\nu_r^2 = (1-\nu_r)\nu_{r-1}^2 + \lambda\nu_k/(\lambda+\alpha)$, and compute
    $$y_r = w_r + \left(\frac{\nu_{r-1}(1-\nu_{r-1})}{\nu_r + \nu_{r-1}^2}\right)(w_r - w_{r-1}). \qquad (5.3)$$
end for
Output: $w_R$.
```

Algorithm 15 `AMD-SVRP`: Accelerated Minibatch Distributed SVR Proximal Iterations.

```
Initialize $w_0, y_0$, $\nu_0 = \sqrt{\lambda/(\lambda+\alpha)}$.
**Sampling** For each machine, sampling $b$ items from $[n]$, they jointly form form a minibatch $\bar{B}$ of size $mb$.
**for** $r = 1, 2, \ldots, R$ **do**
    $\widetilde{x}_0 \leftarrow y_{r-1}$.
    **for** $k = 1, 2, \ldots, K$ **do**
        All machines perform one round of communication to compute the average gradient
            $$\widetilde{v} \leftarrow \frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n}\nabla\ell_{ij}(\widetilde{x}_{k-1}).$$
        $x_0, z_0 \leftarrow \widetilde{x}_{k-1}$.
        **for** $t = 1, 2, \ldots, T$ **do**
            1. Each machine $j$ draws a minibatch $I_t^{(j)}$ of $b$ samples from the local data set, compute and communicate $\frac{1}{b}\sum_{i\in I_t^{(j)}}\nabla\ell_{ij}(z_{t-1})$.
            2. **Find $x_t$ that approximately solve** (5.4) (**Option-I**) or (5.5) (**Option-II**) such that $\widetilde{f}_t(w_t) - \min_w \widetilde{f}_t(w) \le \varepsilon$ (Using DSVRG).
            $$\textbf{Option-I}: \widetilde{f}_t(w) := \frac{1}{mb}\sum_{i\in\bar{B}}\ell_i(w) - \left\langle \frac{1}{mb}\sum_{j=1}^{m}\sum_{i\in I_t^{(j)}}\nabla\ell_{ij}(z_{t-1}), w\right\rangle + \frac{\alpha}{2}||w - y_{r-1}||^2 + \left\langle \frac{\eta}{mb}\sum_{j=1}^{m}\sum_{i\in I_t^{(j)}}\nabla\ell_{ij}(z_{t-1}) + \eta\widetilde{v} - \frac{\eta}{mb}\sum_{j=1}^{m}\sum_{i\in I_t^{(j)}}\nabla\ell_{ij}(\widetilde{x}_{r-1}), w\right\rangle + \frac{\widetilde{\lambda}}{2}||w - z_{t-1}||^2, \quad (5.4)$$
            $$\textbf{Option-II}: \widetilde{f}_t(w) := \frac{1}{2}(w - z_{t-1})^\top\left(\frac{1}{mb}\sum_{i\in\bar{B}}\nabla^2\ell_i(z_{t-1})\right)(w - z_{t-1}) + \frac{\alpha}{2}||w - y_{t-1}||^2 + \left\langle \frac{\eta}{mb}\sum_{j=1}^{m}\sum_{i\in I_t^{(j)}}\nabla\ell_{ij}(z_{t-1}) + \eta\widetilde{v} - \frac{\eta}{mb}\sum_{j=1}^{m}\sum_{i\in I_t^{(j)}}\nabla\ell_{ij}(\widetilde{x}_{r-1}), w\right\rangle + \frac{\widetilde{\lambda}}{2}||w - z_{t-1}||^2. \quad (5.5)$$
            3. Master update
                $$z_t = x_t + \beta_t(x_t - x_{t-1}),$$
            and broadcast $z_t$.
        **end for**
        Update $\widetilde{x}_k \leftarrow x_T$.
    **end for**
    Update $w_r \leftarrow \widetilde{x}_K$.
    Compute $\nu_r \in (0,1)$ such that $\nu_r^2 = (1-\nu_r)\nu_{r-1}^2 + \lambda\nu_k/(\lambda+\alpha)$, and compute
        $$y_r = w_r + \left(\frac{\nu_{r-1}(1-\nu_{r-1})}{\nu_r + \nu_{r-1}^2}\right)(w_r - w_{r-1}). \quad (5.6)$$
**end for**
**Ensure:** $w_R$.
```

step, we are in fact running a minibatch SVRG algorithm in a preconditioned space:

$$f(\widetilde{H}_{\widetilde{\lambda}}^{-1/2} w) = \frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} f_{ij}(\widetilde{H}_{\widetilde{\lambda}}^{-1/2} w), \tag{5.7}$$

where $\widetilde{H}_{\widetilde{\lambda}} = \frac{1}{mb} \sum_{i \in \bar{B}} \nabla^2 \ell_i(w^*) + \widetilde{\lambda} I$ is a sub-sampled Hessian matrix using mb samples. This preconditioning transformation improves the condition number. In order to provide rigorous analysis, we need the following notion of effective dimension and statistical leverage [Zhang, 2005, Hsu et al., 2014, Wang and Zhang, 2017].

Definition 44. *(**Effective dimension at** $\widetilde{\lambda}$) Let the $\lambda_1, ..., \lambda_d$ be the top-d eigenvalues of $H_0 = (1/mn) \sum_{j=1}^{m} \sum_{i=1}^{n} \nabla^2 f_{ij}(w^*)$. The effective dimension $d_{\widetilde{\lambda}}$ (for some $\widetilde{\lambda} \geq 0$) of H_0 is defined as $d_{\widetilde{\lambda}} = \sum_{j=1}^{d} \frac{\lambda_j}{\lambda_j + \widetilde{\lambda}}$.*

The effective dimension $d_{\widetilde{\lambda}}$ is a monotonically decreasing function with respect to $\widetilde{\lambda}$. Thus we always have $d_{\widetilde{\lambda}} \leq d$. Moreover, when $\widetilde{\lambda}$ is not too small and H_0 has a fast decaying spectrum, the effective dimension $d_{\widetilde{\lambda}}$ can be much smaller than d, as demonstrated in the following proposition.

Proposition 45. *We have the following upper bound for $d_{\widetilde{\lambda}}$:*

$$d_{\widetilde{\lambda}} \leq \min_k \left\{ k + \frac{\sum_{j>k} \lambda_j}{\widetilde{\lambda}} \right\}.$$

As an example of a data with fast decaying spectrum, consider generalized linear predictors in a Hilbert space induced by a Gaussian kernel $K(a, a') = \exp(-c||a - a'||^2)$ where c is a bandwidth parameter. The covariance matrix has exponentially decaying eigenvalues with $\lambda_j \leq c_1 \exp(-c_2 j^2)$, where c_1 and c_2 are constants [Zhang et al., 2015]. Using the above proposition, we have that $d_{\widetilde{\lambda}} \leq \mathcal{O}(\sqrt{\log(1/\widetilde{\lambda})})$ even though Gaussian kernel maps the original data to an infinite dimensional space.

The following notion of statistical leverage measures the ratio between the maximum norm and average norm of the whitened data set.

Definition 46. *(**Statistical leverage at** $\widetilde{\lambda}$) Let $H_{\widetilde{\lambda}}(w^*) = (1/mn)\sum_{j=1}^{m}\sum_{i=1}^{n}\nabla^2\ell_{ij}(w^*) + \widetilde{\lambda}I$. The statistical leverage of a data matrix X is bounded by $\rho_{\widetilde{\lambda}}$ at $\widetilde{\lambda}$ if $\forall i \in [n], j \in [m]$, we have:*

$$\frac{||H_{\widetilde{\lambda}}^{-1/2}\ell''_{ij}(w^*)^{1/2}a_{ij}||}{\sqrt{(1/mn)\sum_{j=1}^{m}\sum_{i=1}^{n}||H_{\widetilde{\lambda}}^{-1/2}\ell''_{ij}(w^*)^{1/2}a_{ij}||^2}} \leq \rho_{\widetilde{\lambda}}.$$

In the definition of statistical leverage, the numerator is the maximum norm of the $\widetilde{\lambda}$-whitened data set, while the denominator is its average norm. When rows of a data matrix X are drawn from a sub-Gaussian distributions, the statistical leverage only grows logarithmically with the dimension. We will use the above two quantities to establish the efficiency guarantees for the proposed `AMD-SVRP` algorithm. The following proposition upper bounds the quantity $\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}}$, which is used extensively in the analysis afterwards.

Proposition 47. *Let L be the smoothness parameter for each loss function ℓ. Then $\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} \leq \frac{L}{\widetilde{\lambda}}$.*

The following proposition, adopted from Lemma 8 and Lemma 9 of Wang and Zhang [2017], states how the condition number improves by using a preconditioning matrix.

Proposition 48. *(**Improved condition number after preconditioning**) If we choose $\widetilde{\lambda} = \max\{\lambda, L/\bar{b}\}$ where $\bar{b} \leq mb$, then the condition number of (5.7) can be upper bounded by*

$$\max\left\{4\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}}, \frac{4L\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}}}{\lambda\bar{b}}\right\}$$

with with probability at least $1-\delta$ if $mb = \Omega(\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} \log^2(d/\delta))$.

In Algorithm 15, each step involves solving a subproblem (equations (5.4) or (5.5)), which incorporates higher-order information from mb instances (each machine contributes b samples). We can use a distributed optimization algorithm to solve these subproblems to high accuracy. To that end, we set $\widetilde{\lambda}$ to be large enough so that (5.4) or (5.5) can be solved efficiently. In particular, we use the `DSVRG` algorithm with $\widetilde{\lambda} = L/b$, which allows us to solve (5.4) or (5.5) to ϵ sub-optimality using $\mathcal{O}(b \log(1/\epsilon))$ parallel computations and $\mathcal{O}(\log(1/\epsilon))$ communication rounds.

Based on the above discussion, we have the following communication and computation efficiency guarantees for the `AMD-SVRP` method (Algorithm 15).

Theorem 49. *Suppose the objective in* (5.1) *is quadratic and satisfies Assumption 38. The* `AMD-SVRP` *method (Algorithm 15), with parameters α, b, and $\widetilde{\lambda} = L/b$, finds a solution $\widehat{w}$ using*

$$\widetilde{\mathcal{O}}\left(\sqrt{\frac{\lambda+\alpha}{\lambda}} \cdot \left(1 + \max\left\{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} L}{(\lambda+\alpha)mb^2}, \sqrt{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} L}{(\lambda+\alpha)b}}\right\}\right)\right)$$

rounds of communication and

$$\widetilde{\mathcal{O}}\left(\sqrt{\frac{\lambda+\alpha}{\lambda}} \cdot \left(n + \max\left\{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} L}{(\lambda+\alpha)mb}, \sqrt{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} bL}{\lambda+\alpha}}\right\}\right)\right)$$

parallel computations which satisfies $f(\widehat{w}) - f(w^*) \leq \epsilon$.

Based on Theorem 15, we have the following corollary which states the efficient guarantees by choosing α and b appropriately.

Corollary 50. *Suppose the conditions of Theorem 49 are satisfied, and the parameters in* `AMD-SVRP` *method (Algorithm 15) are set as $\alpha \asymp \frac{L}{mn}, b \asymp \sqrt{\frac{n}{m}}$ and $\widetilde{\lambda} = \frac{L}{b}$, then it finds a* *solution $\widehat{w}$ using*

$$\widetilde{\mathcal{O}}\left(\left(\sqrt{mn}+\sqrt{\min\left\{1,\sqrt{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} m}{n}}\right\}\frac{L}{\lambda}}\right)\right)$$

rounds of communication and

$$\widetilde{\mathcal{O}}\left(\left(n+\sqrt{\frac{nL}{m\lambda}}\right)\right)$$

parallel computations which satisfies $f(\widehat{w}) - f(w^*) \leq \epsilon$.

Corollary 50 tells us that Algorithm 15 always obtains near-linear speed up in computation compared to a single machine, but with potential of using fewer communication rounds than Algorithm 13, if $\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}}$ is small.

Table 5.1 summarizes the theoretical communication and computation efficiency guarantees for various distributed optimization algorithms in solving finite-sum problems.[2]. We highlight the following conclusions from the comparisons:

i) In terms of both communication and computation, accelerated methods are preferred over non-accelerated methods (`CoCoA+` and `DANE`) as they only have square root dependency on the condition number L/λ. Among the accelerated methods, we can conclude that `DASVRG` improves over `DiSCO/AIDE` (by a factor of $n^{1/4}$), and `DiSCO/AIDE` improves over `AccGD` and `ADMM` (also by a factor of $n^{1/4}$).

ii) `AMD-SVRG/AMD-SVRP` algorithms are the only ones that achieve near-linear speed up in computation, thus matching the computational lower bound in distributed finite-sums optimization.

iii) In terms of communication, `AMD-SVRP` improves over `AMD-SVRG` when the effective dimension and statistical leverage is small. However, it is still worse than `DASVRG`, which matches the communication lower bound.

2. In the communication lower bound we implicitly assume $L/\lambda <= n^3$, otherwise each machine can just send their whole local data and the lower bound becomes invalid.

Table 5.2: List of data sets used in the experiments.

Name	#Instances	#Features	Task
aloi	108,000	128	Regression
cadata	20,640	8	Regression
codrna	59,535	8	Classification
covtype	581,012	54	Classification
synthetic-c	500,000	500	Classification
magic04	19,020	10	Classification
synthetic-r	500,000	500	Regression
spambase	4,601	56	Classification
svmguide1	7,089	4	Classification
tomshardware	28,179	96	Regression
twitter	58,3250	77	Regression
year	463,715	90	Regression

5.4 Experiments

We compare the proposed `AMD-SVRG`/`AMD-SVRG-P` and `AMD-SVRP` algorithms to existing distributed `SVRG` approaches for minimizing (5.1). We tested these algorithms on 6 classification and 6 regression data sets, which were summarized in Table 5.2. Most of the data sets involve real-world classification or regression tasks, with the source available from the LibSVM website[3]. Besides real world data, we also consider a synthetic data set for each task, where each data point $\{a_i, b_i\}_{i=1}^n$ is drawn i.i.d. from the following model:

$$\text{Regression}: \quad b_i = \bar{w}^\top a_i + \varepsilon_i, \quad a_i \sim \mathcal{N}(0, \Sigma), \quad \varepsilon_i \sim \mathcal{N}(0,1), \quad \forall i \in [n],$$

$$\text{Classification}: \quad P(b_i = \pm 1) = \frac{\exp(b_i \bar{w}^\top a_i)}{1 + \exp(b_i \bar{w}^\top a_i)}, \quad x_i \sim \mathcal{N}(0, \Sigma), \quad \forall i \in [n],$$

and each entry of $\bar{w}$ is drawn i.i.d. from $\mathcal{N}(0,1)$. We set $\Sigma_{ij} = 2^{-|j-k|/500}, \forall j,k \in [n]$ to make the problem ill-conditioned with fast decaying spectrum.

We consider ridge regression, for which $\ell(w^\top a, b) = (1/2)(\ell(w^\top a - b)^2$, and ℓ_2 regularized logistic regression, for which $\ell(w^\top a, b) = \log(1 + \exp(-b(w^\top a)))$, objectives for regression

3. `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

and classification problems, respectively. We normalize the data set by $a_i \leftarrow a_i/(\max_i ||a_i||^2)$ to ensure that the maximum norm of a data point is 1, and set the regularization parameter λ to be $10^{-2}/N$ where N is the total number of data instances. With these parameters, the condition number L/λ is much larger than the total sample size, which represents the typical situation of "large condition number" regime where existing DSVRG/DASVRG algorithms are not computationally optimal. We vary the number of machines $m = 16$ and $m = 64$ and partition data uniformly across the machines, so that each machine gets access to $n = N/m$ instances.

We compare the proposed AMD-SVRG/AMD-SVRG-P and AMD-SVRP methd with DSVRG approach and its accelerated version DASVRG [Lee et al., 2017a], as these algorithms dominate other previously proposed methods in both communication and computation efficiency. See Table 5.1 and also Lee et al. [2017a]. For all algorithms, we compare both the communication and (parallel) computation cost. In particular, we use the number of (parallel) gradient evaluations each machine performed to evaluate the computation cost, since gradient computations are the main computational part in these algorithms. Moreover, we use the rounds of communication to measure the communication cost, where at each round each machine is allowed to broadcast and receive one vector of the dimension as a data point x_{ij}. For the proposed AMD-SVRP method, at every iteration we simply run one pass of DSVRG method initialized with y_{t-1}, to approximately solve (5.4). For AMD-SVRG and AMD-SVRP methods, following the theoretical analysis, we set the minibatch size each machine process to be $\lfloor\sqrt{n/m}\rfloor$ and run $T = \lfloor\sqrt{L/(\lambda b)}\rfloor$ iterations. For DSVRG and DASVRG methods, at each stage we first compute the full gradient in parallel, then randomly pick one machine, and perform stochastic variance reduced gradient update on that machine's whole local data set. For all accelerated methods (all compared approaches except DSVRG), we set the number of reference full gradient computations within each acceleration stage to be $K = 2$. For AMD-SVRG-P method, we set $b = \lfloor\sqrt{n/m}\rfloor$ and run $T = \lfloor n/b\rfloor$ iterations, which we found to achieve a good balance between communication and computation. Section 5.4.1 explore the trade-offs between communication and computation in `AMD-SVRG-P`.

The results are shown in Figure 5.1 to Figure 5.4. We plot the communication and computation cost for each algorithm to find a solution that achieves 1×10^{-6} primal objective suboptimality. We have the following observations:

- In terms of parallel computation cost, we observe that the proposed `AMD-SVRG` and `AMD-SVRP` methods significantly improve over the `DSVRG` and `DASVRG` methods. In many cases, the computational speedup over `DSVRG/DASVRG` is often at the scale of $10\times$ to $100\times$. Furthermore, we observe significant computational advantages for the proposed methods when we have more machines (compare the right column with the left). This is consistent with the theoretical analysis (Table 5.1).

- In terms of the communication cost, `AMD-SVRG` and `AMD-SVRP` methods are obviously worse than `DSVRG/DASVRG`. But for all methods, the number of bits each machine communicated is still of a much smaller order compared with the number of floating point operations. Thus for many modern distributed computing platforms, the communication cost of these algorithms might still play a relatively minor role in affecting the overall runtime.

- When we compared `AMD-SVRG` and `AMD-SVRP`, we observed that `AMD-SVRP` method often uses less communication, but with comparable or fewer computation resources. This demonstrates the effectiveness of proximal iterations to incorporate higher-order information in a minibatch. The practical variant `AMD-SVRG-P` seems to have significantly less communication cost, but comparable computation with `AMD-SVRG`, which demonstrate that the "sequential then average" strategy works better than the standard minibatch approach. This is consistent with previous empirical studies [Konečný et al., 2016].

5.4.1 Communication-computation trade-offs in `AMD-SVRG-P`

In the `AMD-SVRG-P` algorithm, the minibatch size per machine b can be interpreted as a parameter to trade-off between communication and computation. When b is relatively small, then the effect of local sequential updates is small and the method is closely connected to `AMD-SVRG` (when $b = 1$, then it is exactly `AMD-SVRG`, but with a suboptimal minibatch size), which is computationally optimal, but with a potentially heavy communication. When b is large, then the method is similar to `DASVRG`, which is communication optimal, but with a potentially heavy computation. The difference here is that `AMD-SVRG-P` runs multiple stochastic sequential updates then takes average and also uses a momentum scheme. In this section, we explore these trade-offs via experimental studies. In particular we vary the parameter b in $\{\lceil n/2^i \rceil\}$ where $i \in \{0, 2, 4, 6, 8, 10\}$ and set $T = \lfloor n/b \rfloor$.

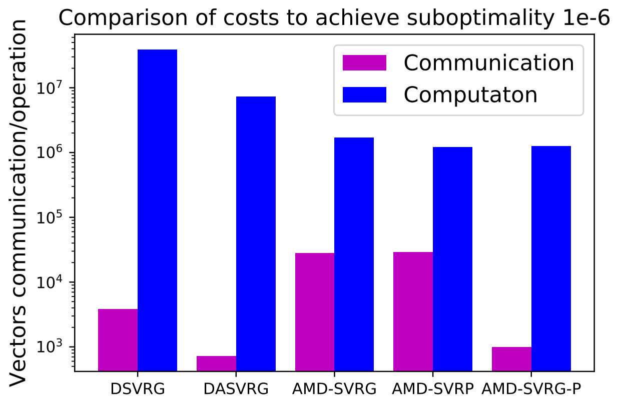
The results are plotted in Figure 5.5. We observe that as b decreases, the communication cost increases significantly. At the same time, the computation cost first decreases and then does not change too much since it already achieves near-linear computation speedup. From the plots we observe that by choosing $b = \lceil n/4 \rceil$ or $\lceil n/16 \rceil$ the algorithms obtains a good balance between computation and communication.

5.5 Proofs of technical results

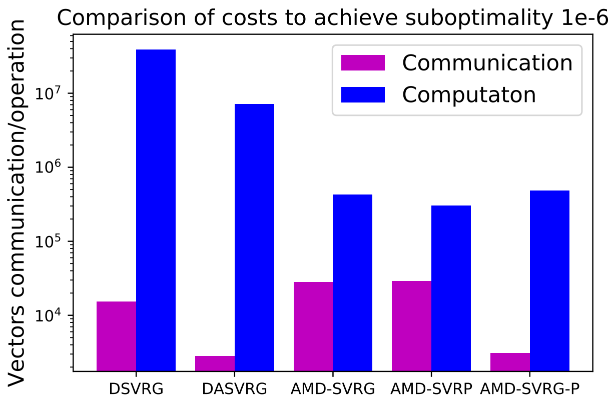
5.5.1 Proof of Corollary 40

Proof. Let G_j to be the total number of individual function gradient calls that machine j during the the optimization process to achieve ϵ-objective suboptimality. By the oracle complexity of first-order optimization methods (Theorem 8 of [Woodworth and Srebro, 2016]), the total number or gradient oracle calls for all machines to make must obey the following inequality:

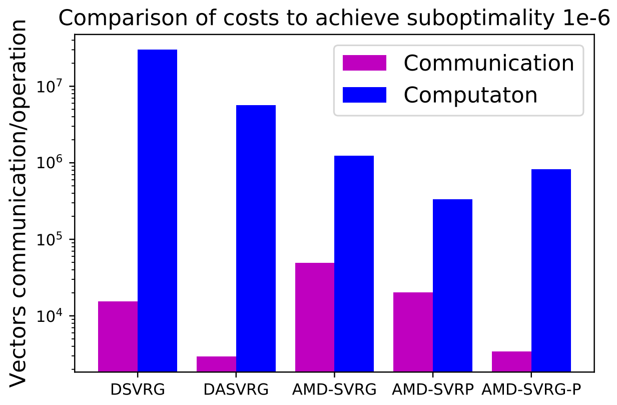
$$\sum_{j=1}^{m} G_j \geq \Omega\left(mn + \sqrt{\frac{mnL}{\lambda}} \log\left(\frac{1}{\epsilon}\right)\right),$$



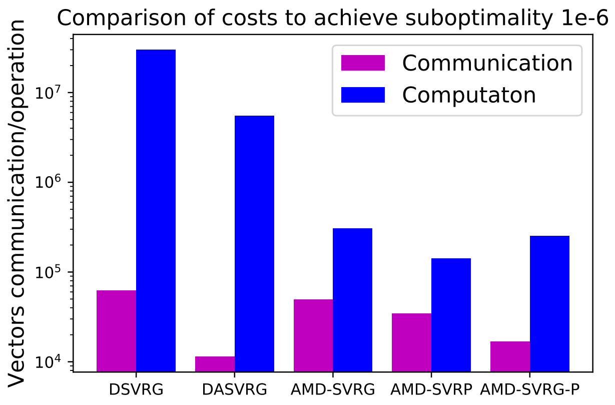
aloi, $m = 16$



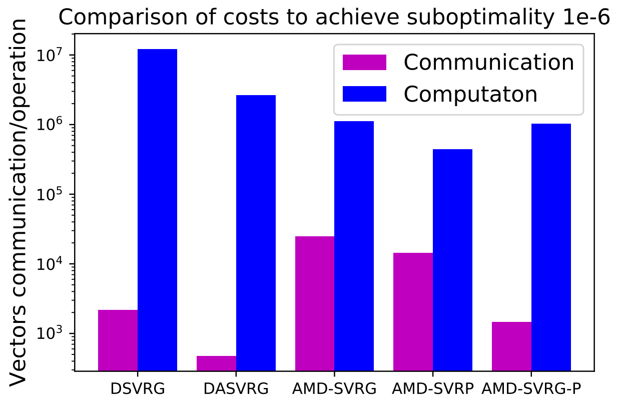
aloi, $m = 64$



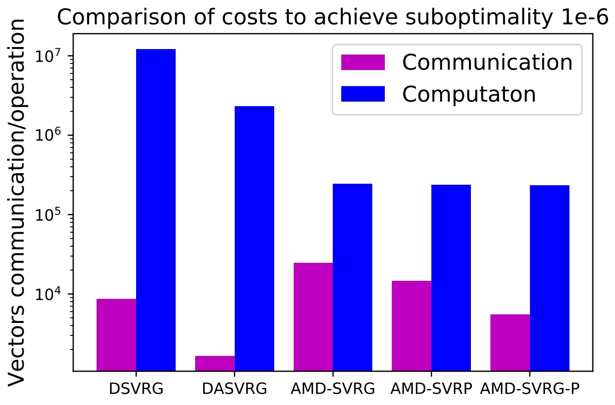
cadata, $m = 16$



cadata, $m = 64$

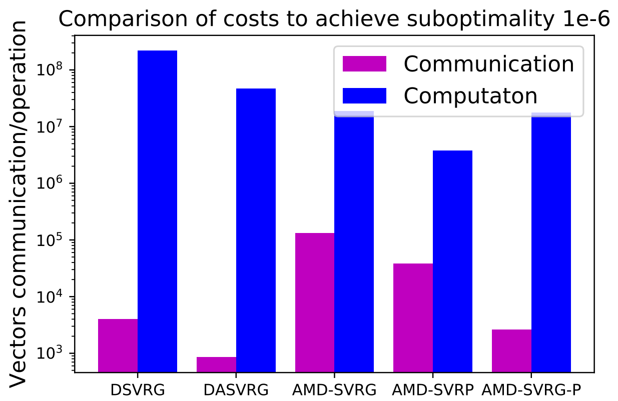


codrna, $m = 16$

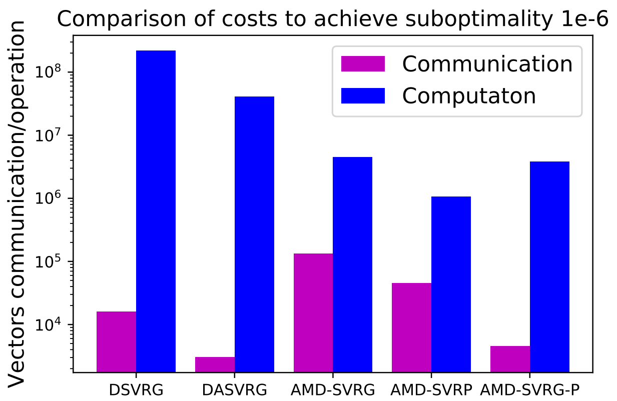


codrna, $m = 64$

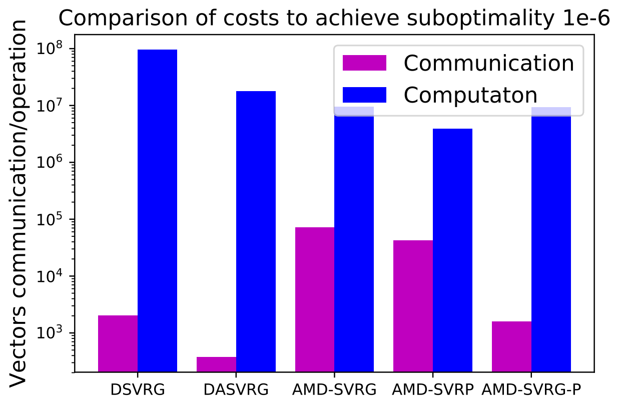
Figure 5.1: Comparison of communication and computation cost of various algorithms for distributed optimization.



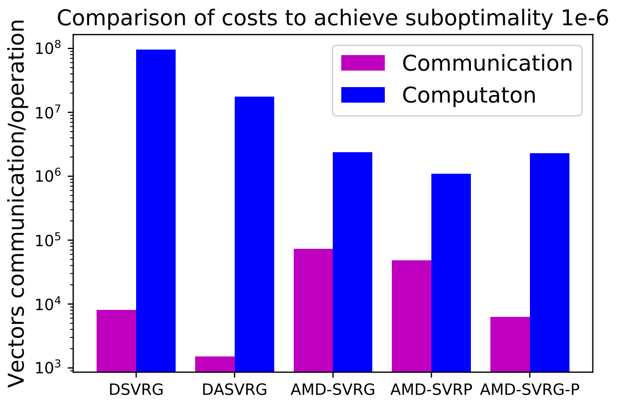
covtype, $m = 16$



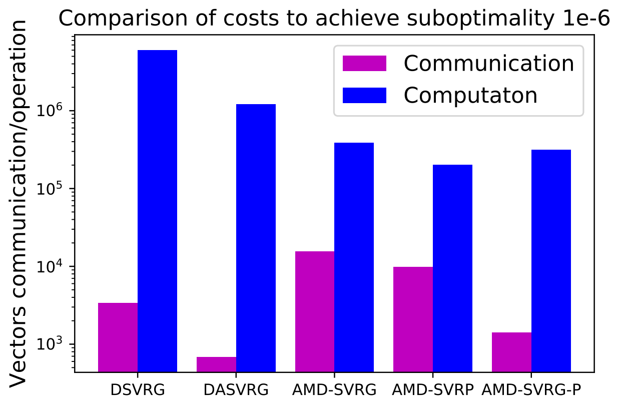
covtype, $m = 64$



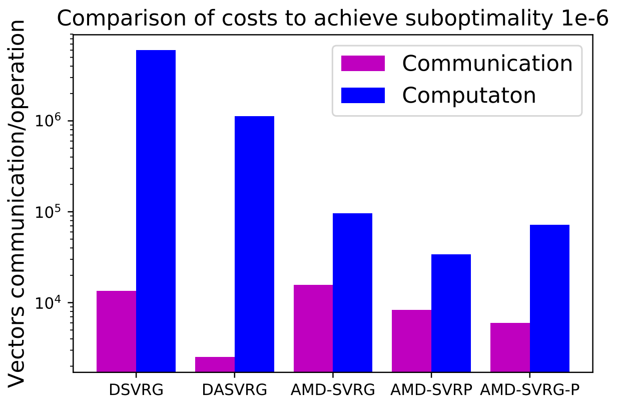
synthetic-c, $m = 16$



synthetic-c, $m = 64$

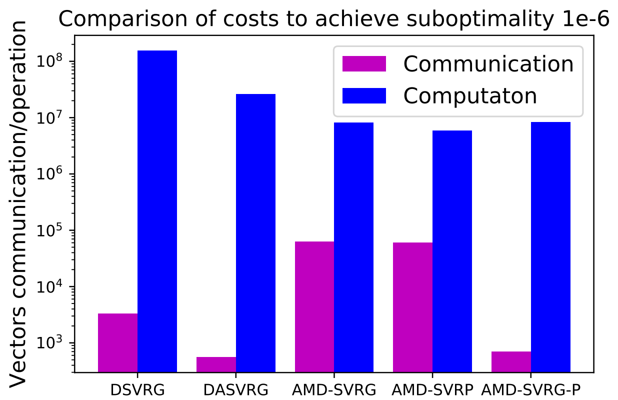


magic04, $m = 16$

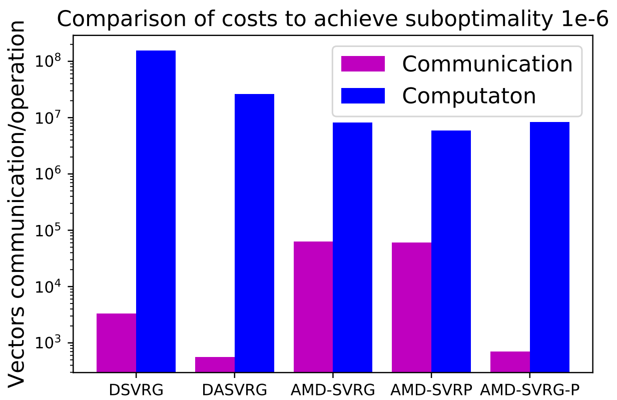


magic04, $m = 64$

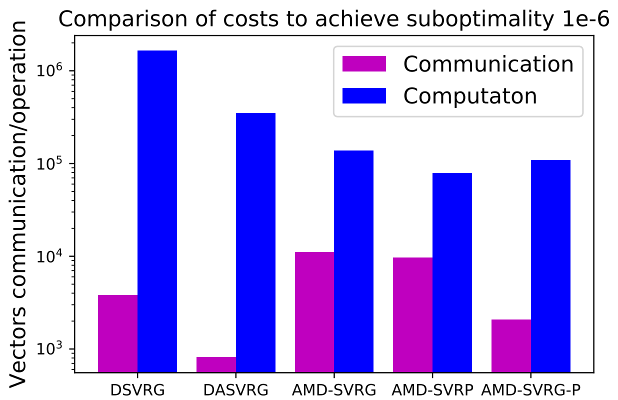
Figure 5.2: Comparison of communication and computation cost of various algorithms for distributed optimization (cont.).



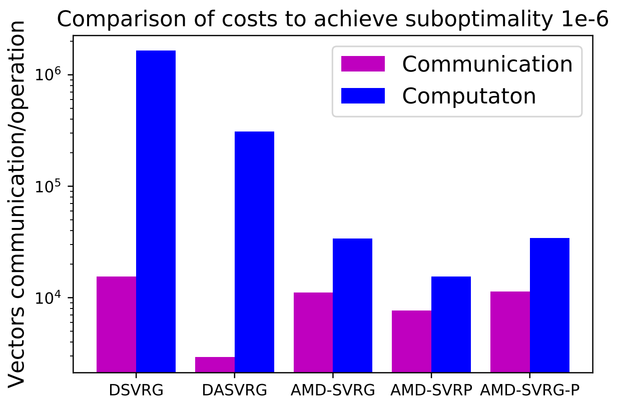
synthetic-r, $m = 16$



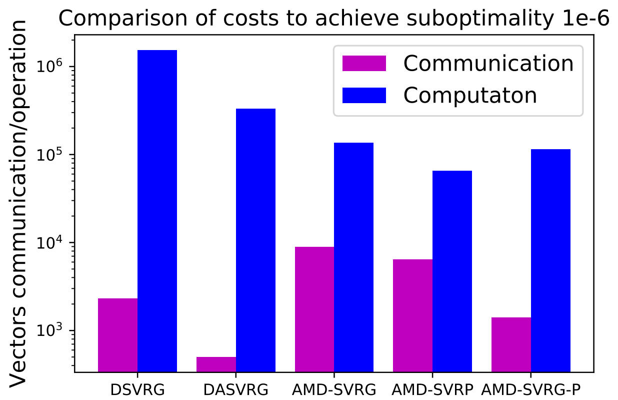
synthetic-r, $m = 64$



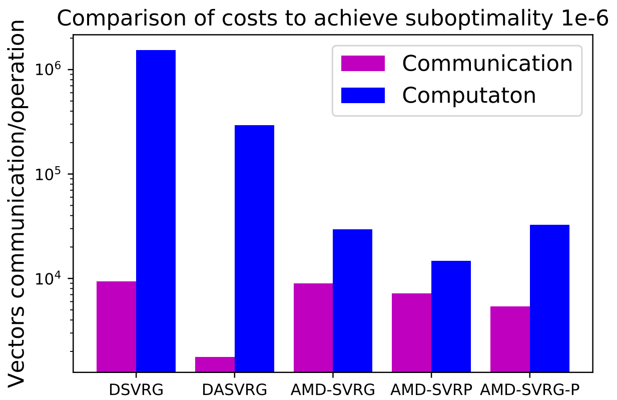
spambase, $m = 16$



spambase, $m = 64$

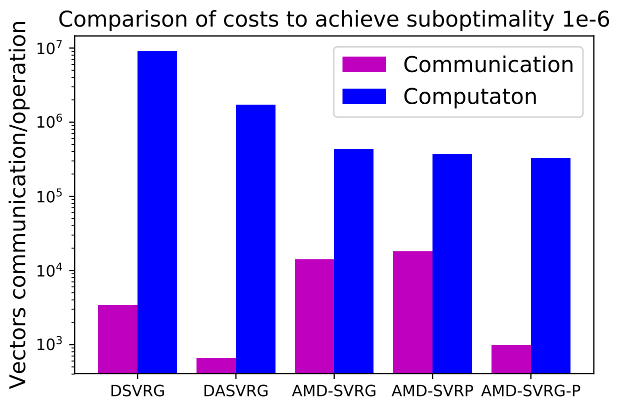


svmguide1, $m = 16$

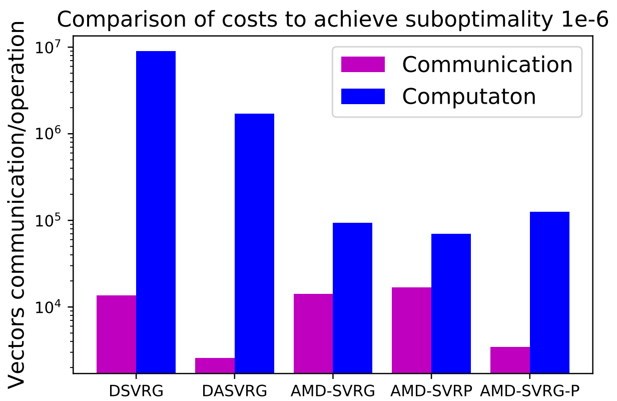


svmguide1, $m = 64$

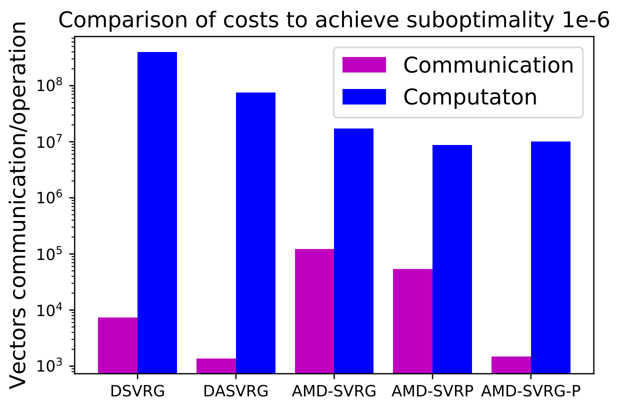
Figure 5.3: Comparison of communication and computation cost of various algorithms for distributed optimization (cont.).



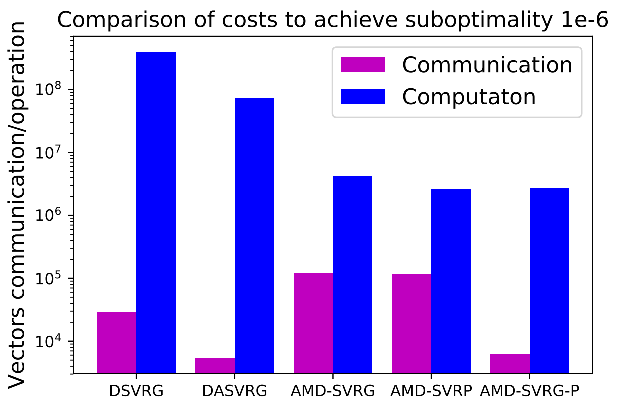
tomshardware, $m = 16$



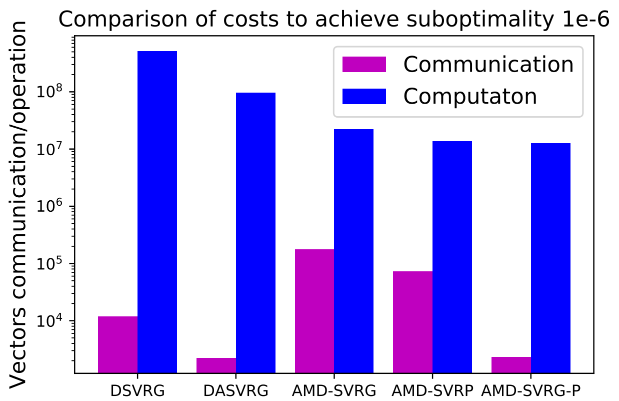
tomshardware, $m = 64$



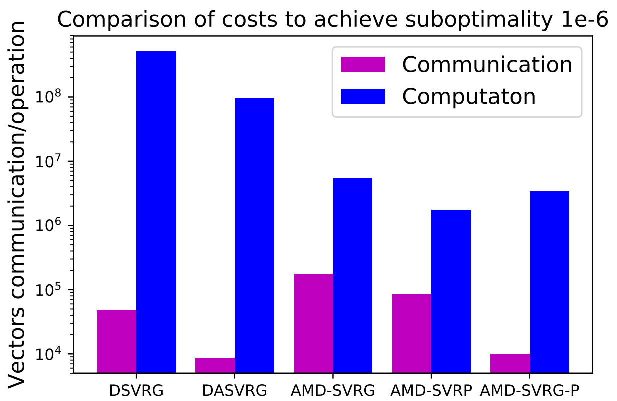
twitter, $m = 16$



twitter, $m = 64$

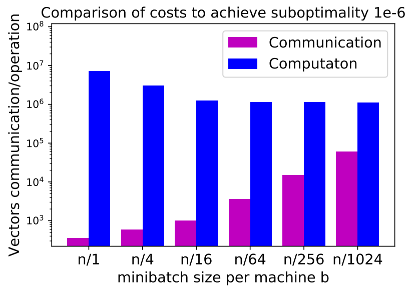


year, $m = 16$

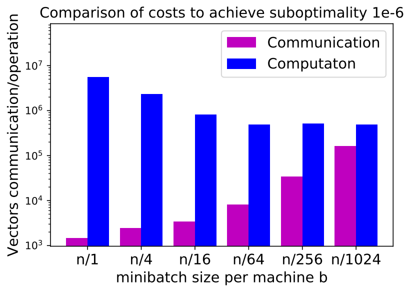


year, $m = 64$

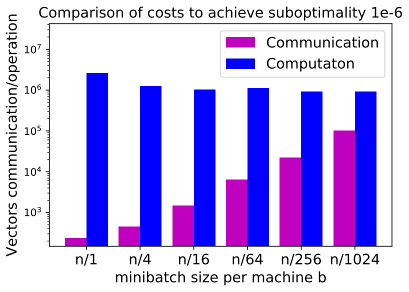
Figure 5.4: Comparison of communication and computation cost of various algorithms for distributed optimization (cont.).



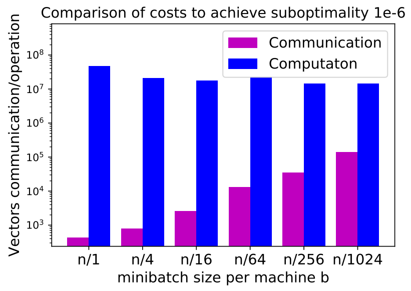
aloi



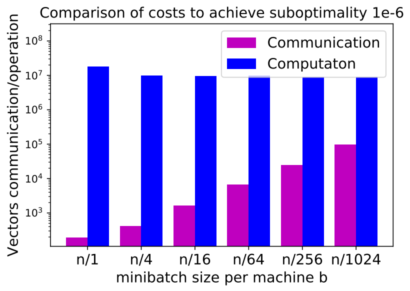
cadata



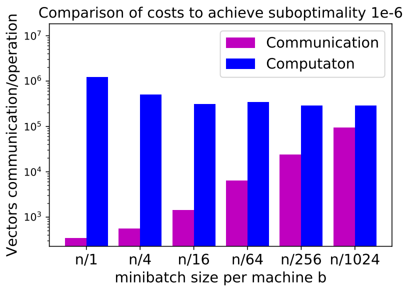
codrna



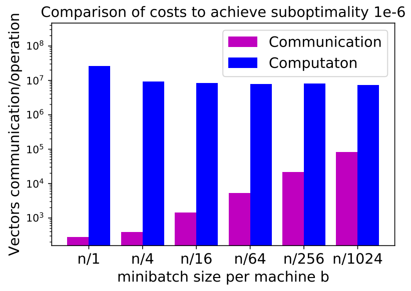
covtype



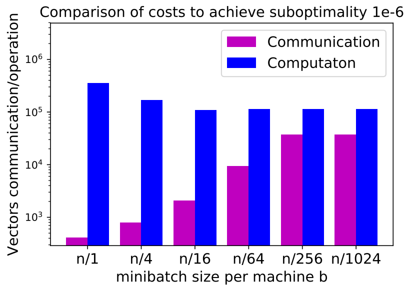
synthetic-c



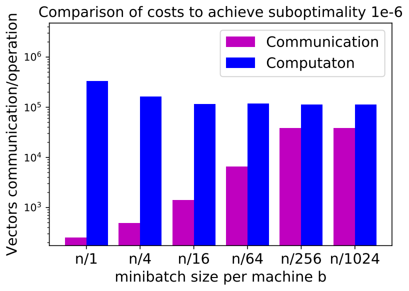
magic04



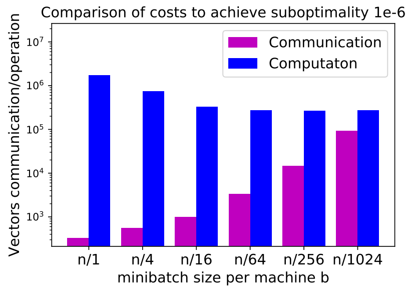
synthetic-r



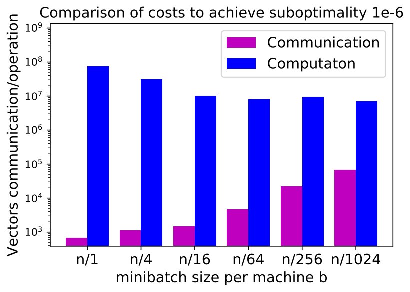
spambase



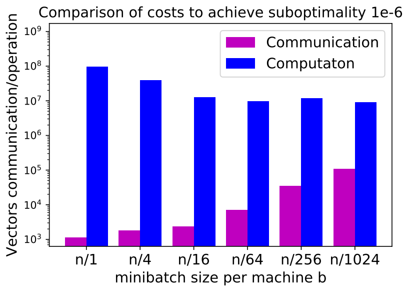
svmguide1



tomshardware



twitter



year

Figure 5.5: Experiments on the communication-computation trade-offs in `AMD-SVRG-P` Algorithm, the x-axis represents different configurations of b.

which implies $\max_{j\in[m]} G_j \geq \Omega\left(n + \sqrt{nL/(m\lambda)}\log(1/\epsilon)\right)$. While in parallel implementation, the paralleled number of gradient oracle calls must be at least $\max_{j\in[m]} G_j$, which concludes the proof. □

5.5.2 Proof of Theorem 41

Proof. Fist using the theory of accelerated proximal point algorithm (Lemma 51), we know the number of outer rounds (indexed by r) in Algorithm 13 are upper bounded by

$$R \leq \mathcal{O}\left(\sqrt{\frac{\lambda+\alpha}{\lambda}}\log\left(\frac{1}{\epsilon}\right)\right).$$

Further more, each proximal point problem (5.2) is now L-smooth and $(\lambda+\alpha)$-strongly convex, by the iteration complexity of accelerated minibatch `SVRG` (Lemma 52), since the effective minibatch size in Algorithm 13 is mb, the number of stochastic iterations at each stage (indexed by t) are bounded by

$$T \leq \mathcal{O}\left(1 + \max\left\{\frac{L}{(\lambda+\alpha)mb}, \sqrt{\frac{L}{(\lambda+\alpha)}}\right\}\right).$$

By choosing T as above, the number of middle loops (indexed by k) is upper bounded by $K \leq \mathcal{O}(\log(1/\epsilon))$. Combining above we know the total rounds of communication can be bounded by

$$R\cdot K\cdot T \leq \mathcal{O}\left(\sqrt{\frac{\lambda+\alpha}{\lambda}}\cdot\left(1+\max\left\{\frac{L}{(\lambda+\alpha)mb}, \sqrt{\frac{L}{\lambda+\alpha}}\right\}\right)\cdot\log^2\left(\frac{1}{\epsilon}\right)\right).$$

And the total paralleled computation can be upper bounded by

$$R\cdot K\cdot n + R\cdot K\cdot T\cdot b \leq \mathcal{O}\left(\sqrt{\frac{\lambda+\alpha}{\lambda}}\cdot\left(n+\max\left\{\frac{L}{(\lambda+\alpha)m}, b\sqrt{\frac{L}{\lambda+\alpha}}\right\}\right)\cdot\log^2\left(\frac{1}{\epsilon}\right)\right)$$

□

5.5.3 Proof of Corollary 42

Proof. Simply substitute the order of b and α stated in Corollary 42 to Theorem 41, we obtain the stated result. □

5.5.4 Proof of Corollary 43

Proof. We follow the `AdaptReg` strategy in [Allen-Zhu and Hazan, 2016] by adding a $\frac{\sigma_t}{2}||x - x_t||$ regularization at each stage, where $\sigma_s = (f(x_0) - f(x^*))/(||x_0 - x^*||^2 2^s)$. At each stage s, we are required to reduce the primal objective suboptimality be a factor of 4. Let $S = \log(f(x_0) - f(x^*)/\epsilon)$, by Theorem 3.1 of [Allen-Zhu and Hazan, 2016], and combine the results with Corollary 42. We know the total number of communication rounds for `AMD-SVRG` is

$$\widetilde{\mathcal{O}}\left(\sum_{s=1}^{S}\left(\sqrt{mn} + \sqrt{\frac{L}{\sigma_s}}\right)\right) = \widetilde{\mathcal{O}}\left(\sqrt{mn} + B\sqrt{\frac{L}{\epsilon}}\right),$$

and its paralleled computation cost is upper bounded by

$$\widetilde{\mathcal{O}}\left(\sum_{s=1}^{S}\left(n + \sqrt{\frac{nL}{m\sigma_s}}\right)\right) = \widetilde{\mathcal{O}}\left(\sqrt{mn} + B\sqrt{\frac{nL}{m\epsilon}}\right).$$

□

5.5.5 Proof of Proposition 45

Proof. We first decompose the terms $\frac{\lambda_j}{\lambda_j + \widetilde{\lambda}}$ into two parts, the first one contains indexes smaller or equal than k and the second part contains indexes larger than k, then we upper bound these two parts get obtain the stated results.

$$
\begin{aligned}
d_{\widetilde{\lambda}} = & \sum_j \frac{\lambda_j}{\lambda_j + \widetilde{\lambda}} = \min_k \left\{ \sum_{j \le k} \frac{\lambda_j}{\lambda_j + \widetilde{\lambda}} + \sum_{j > k} \frac{\lambda_j}{\lambda_j + \widetilde{\lambda}} \right\} \\
& \le \min_k \left\{ \sum_{j \le k} \frac{\lambda_j + \widetilde{\lambda}}{\lambda_j + \widetilde{\lambda}} + \sum_{j > k} \frac{\lambda_j}{\widetilde{\lambda}} \right\} = \min_k \left\{ k + \frac{\sum_{j>k} \lambda_j}{\widetilde{\lambda}} \right\}
\end{aligned}
$$

□

5.5.6 *Proof of Proposition 47*

Proof. First noted that

$$
\begin{aligned}
d_{\widetilde{\lambda}} = & \sum_{j=1}^{d} \frac{\lambda_j}{\lambda_j + \widetilde{\lambda}} = \operatorname{trace}\left(H_{\widetilde{\lambda}}(w^*)^{-1} H_0 \right) \\
& = \operatorname{trace}\left(\left(\frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} \ell_{ij}''(w^*) a_{ij} a_{ij}^\top \right) H_{\widetilde{\lambda}}(w^*)^{-1} \right) \\
& = \operatorname{trace}\left(\frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} \ell_{ij}''(w^*) a_{ij}^\top H_{\widetilde{\lambda}}(w^*)^{-1} a_{ij} \right) \\
& = \frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} \ell_{ij}''(w^*) a_{ij}^\top H_{\widetilde{\lambda}}(w^*)^{-1} a_{ij} = \frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} ||H_{\widetilde{\lambda}}(w^*)^{-1/2} \ell_{ij}''(w^*) a_{ij}||^2
\end{aligned}
$$

Thus we have

$$
\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} = \max_{i \in [n] m j \in [m]} ||H_{\widetilde{\lambda}}(w^*)^{-1/2} \ell_{ij}''(w^*) a_{ij}||^2 \le \frac{||\ell_{ij}''(w^*) a_{ij}||}{\widetilde{\lambda}} \le \frac{L}{\widetilde{\lambda}}.
$$

□

5.5.7 Proof of Theorem 49

Proof. By Proposition 48, the condition number of $f(\widetilde{H}_{\widetilde{\lambda}} w)$ for randomized optimization algorithm have been improved to $\mathcal{O}((L\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}})/(\lambda b))$ after preconditioning using mb samples as Algorithm 15 did. Moreover, since we choose $\widetilde{\lambda} \asymp L/b$, the subproblem 5.4 is which involves mb samples has condition number $\mathcal{O}(b)$, thus can be solved to ϵ-accuracy within $\mathcal{O}(\log(1/\epsilon))$ rounds of communication, and $\mathcal{O}(b)$ paralleled computation. Then by the iteration complexity of accelerated minibatch `SVRG`, we know the number of stochastic iterations needed at each stage can be upper bounded by

$$T \leq \mathcal{O}\left(1 + \max\left\{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} L}{(\lambda+\alpha)mb^2}, \sqrt{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} L}{(\lambda+\alpha)b}}\right\}\right),$$

to ensure $K \leq \mathcal{O}(\log(1/epsilon))$. Combining with the accelerated proximal point iteration complexity $R \leq \mathcal{O}\left(\sqrt{(\lambda+\alpha)/\lambda}\log(1/\epsilon)\right)$, we know the communication rounds can be upper bounded by

$$R \cdot K \cdot T \leq \widetilde{\mathcal{O}}\left(\sqrt{\frac{\lambda+\alpha}{\lambda}} \cdot \left(1 + \max\left\{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} L}{(\lambda+\alpha)mb^2}, \sqrt{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} L}{(\lambda+\alpha)b}}\right\}\right)\right),$$

and the paralleled computation cost can ube upper bounded by

$$R \cdot K \cdot n \cdot \log(1/\epsilon) + R \cdot K \cdot T \cdot b \cdot \log(1/\epsilon) \leq \widetilde{\mathcal{O}}\left(\sqrt{\frac{\lambda+\alpha}{\lambda}} \cdot \left(n + \max\left\{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} L}{(\lambda+\alpha)mb}, \sqrt{\frac{\rho_{\widetilde{\lambda}}^2 d_{\widetilde{\lambda}} bL}{\lambda+\alpha}}\right\}\right)\right).$$

□

5.5.8 Proof of Corollary 50

Proof. Substitute the order of b, α and $\widetilde{\lambda}$ stated in Corollary 50 to Theorem 49, and combine with Proposition 47 we obtain the stated result. $\square$

5.5.9 Collections of tools in the analysis

Lemma 51. *(**Iteration complexity of catalyst acceleration**, Theorem 3.1 of [Lin et al., 2015] rephrased) For any λ-strongly convex function $f(w)$, If the minimization step of* (5.2) *satisfies $\forall r$*

$$f(w_r)-\frac{\alpha}{2}||w_r-y_{r-1}||-\min_w\left(f(w)+\frac{\alpha}{2}||w-y_{r-1}||\right)\le\frac{2}{9}(f(w_0)-f(w^*))\left(1-\frac{9}{10}\sqrt{\frac{\lambda}{\lambda+\alpha}}\right)^r,$$

then if we initialize $\nu_0=\sqrt{\frac{\lambda}{\lambda+\alpha}}$ and set ν_r such that $\nu_r^2=(1-\nu_r)\nu_{r-1}^2+(\lambda\nu_r)/(\lambda+\alpha)$, then the sequences $\{w_r\}$ in Algorithm 13 satisfies

$$f(w_r)-f(w^*)\le\frac{800(\lambda+\alpha)}{\lambda}\left(1-\frac{9}{10}\sqrt{\frac{\lambda}{\lambda+\alpha}}\right)^{r+1}(f(w_0)-f(w^*)).$$

Thus after

$$R\le\mathcal{O}\left(\sqrt{\frac{\lambda+\alpha}{\lambda}}\log\left(\frac{1}{\epsilon}\right)\right)$$

stages, we have $f(w_R)-f(w^*)\le\epsilon$.

Lemma 52. *(**Iteration complexity of accelerated minibatch `SVRG`**, Theorem 1 and Corollary 1 of [Nitanda, 2014] rephrased) For any λ-strongly convex and L-smooth function $f(w)$ in form* (5.1)*, If we use accelerated minibatch `SVRG` it with minibatch size b, then it we set the stepsize as*

$$\eta\asymp\min\left\{\frac{b\lambda^2}{L},\frac{1}{L}\right\},$$

then to obtain a solution with ϵ primal objective suboptimality, the following amount of total *gradient calculations are sufficient:*

$$\mathcal{O}\left(\left(mn + \max\left\{\frac{L}{\lambda}, b\sqrt{\frac{L}{\lambda}}\right\}\right)\log\left(\frac{1}{\epsilon}\right)\right).$$

CHAPTER 6
DISTRIBUTED MULTI-TASK LEARNING WITH SHARED SPARSITY

6.1 Motivation

Learning multiple tasks simultaneously allows transferring information between related tasks and for improved performance compared to learning each tasks separately Caruana [1997]. It has been successfully exploited in, for example, spam filtering Weinberger et al. [2009], web search Chapelle et al. [2010], disease prediction Zhou et al. [2013] and eQTL mapping Kim and Xing [2010].

Tasks could be related to each other in a number of ways. In this chapter, we focus on the high-dimensional multi-task setting with joint support where a few variables are related to all tasks, while others are not predictive Turlach et al. [2005], Obozinski et al. [2011], Lounici et al. [2011]. The standard approach is to use the mixed ℓ_1/ℓ_2 or ℓ_1/ℓ_∞ penalty, as such penalties encourage selection of variables that affect all tasks. Using a mixed norm penalty leads to better performance in terms of prediction, estimation and model selection compared to using the ℓ_1 norm penalty, which is equivalent to considering each task separately.

Shared support multi-task learning is generally considered in a centralized setting where

Approach	Communication	ℓ_1/ℓ_2 estimation error	Prediction error
Local lasso	0	$\sqrt{\frac{\lvert S\rvert^2 \log p}{n}}$	$\frac{\lvert S\rvert \log p}{n}$
Group lasso	$\mathcal{O}(np)$	$\frac{\lvert S\rvert}{\sqrt{n}}\sqrt{1+\frac{\log p}{m}}$	$\frac{\lvert S\rvert}{n}\left(1+\frac{\log p}{m}\right)$
DSML	$\mathcal{O}(p)$	$\frac{\lvert S\rvert}{\sqrt{n}}\sqrt{1+\frac{\log p}{m}}+\frac{\lvert S\rvert^2 \log p}{n}$	$\frac{\lvert S\rvert}{n}\left(1+\frac{\log p}{m}\right)+\frac{\lvert S\rvert^3(\log p)^2}{n^2}$

Table 6.1: Comparison of scaling of parameter estimation errors and prediction errors, for well-conditioned subgaussian feature vectors, with m tasks, n samples per task, p features of which $|S|$ are relevant—see Section 6.4 for dependence on other parameters. **DSML improves over Lasso and has the same leading term as the Group lasso as long as $m \lesssim n/(|S|^2 \log p)$.**

data from all tasks are available on a single machine, and the estimator is computed using a standard single-thread algorithm. With the growth of modern massive data sets, there is a need to revisit multi-task learning in a distributed setting, where tasks and data are distributed across machines and communication is expensive. In particular, we consider a setting where each machine holds one "task" and its related data.

We develop an efficient distributed algorithm for multi-task learning that exploits shared sparsity between tasks. Our algorithm (DSML) requires only one round of communication between the workers and the central node, involving each machine sending a vector to the central node and receiving back a support set. Despite the limited communication, our algorithm enjoys the same theoretical guarantees as the centralized approach under mild conditions. This is summarized in Table 6.1, which compares the prediction and parameter error guarantees of the Lasso run locally on each machine, the communication-intensive group Lasso procedure, and our communication-efficient DSML.

6.1.1 Related work on distributed learning and optimization

With the increase in the volume of data used for machine learning, and the availability of distributed computing resources, distributed learning and the use of distributed optimization for machine learning has received much attention.

Most of work on distributed optimization focuses on "consensus problems", where each machine holds a different objective $f_i(\mathbf{w})$ and the goal is to communicate between the machines to jointly optimize the average objective $1/m \sum_i f_i(\mathbf{w})$, that is, to find a *single* vector $\mathbf{w}$ that is good for all local objectives Boyd et al. [2011]. The difficulty of consensus problems is that the local objectives might be rather different, and, as a result, one can obtain lower bounds on the amount of communication that must be exchanged in order to reach a joint optimum. In particular, the problem becomes harder as more machines are involved.

The consensus problem has also been studied in the stochastic setting Ram et al. [2010], in which each machine receives stochastic estimates of its local objective. Thinking of each local objective as a generalization error w.r.t. a local distribution, we obtain the following distributed learning formulation Balcan et al. [2012]: each machine holds a different source distribution $\mathcal{D}_i$ from which it can sample, and this distribution corresponds to a different local generalization error $f_i = \mathbb{E}_{(X,y)\sim\mathcal{D}_i}[loss(\mathbf{w}, X, y)]$. The goal is to find a single predictor $\mathbf{w}$ that minimizes the average generalization error, based on samples sampled at the local nodes. Again, the problem becomes harder when more machines are involved and one can obtain lower bounds on the amount of communication required—Balcan et al. [2012] carry out such an analysis for several hypothesis classes.

A more typical situation in machine learning is one in which there is only a single source distribution $\mathcal{D}$, and data from this single source is distributed randomly across the machines (or equivalently, each machine has access to the same source distribution $\mathcal{D}_i = \mathcal{D}$). Such a problem can be reduced to a consensus problem by performing consensus optimization of the empirical errors at each machine. However, such an approach ignores several issues: first, the local empirical objectives are not arbitrarily different, but rather quite similar, which can and should be taken advantage of in optimization Shamir et al. [2014]. Second, since each machine has access to the source distribution, there is no lower bound on communication—an entirely "local" approach is possible, were each machine completely ignores other machines and just uses its own data. In fact, increasing the number of machines only makes the problem easier (in that it can reduce the runtime or number of samples per machine required to achieve target performance), as additional machines can always be ignored. In such a setting, the other relevant baseline is the "centralized" approach, where all data is communicated to a central machine which computes a predictor centrally. The goal here is then to obtain performance close to that of the "centralized" approach (and much better than the "local" approach), using roughly the same number of samples, but with low communication and computation costs. Such single-source distributed problems have been studied both in terms of predictive performance Shamir and Srebro [2014], Jaggi et al. [2014] and parameter estimation Zhang et al. [2013f,e], Lee et al. [2017b].

In this chapter we suggest a novel setting that combines aspects of the above two settings. On one hand, we assume that each machine has a different source distributions $\mathcal{D}_i(X, y)$, corresponding to a different task, as in consensus problems and in Balcan et al. [2012]. For example, each machine serves a different geographical location, or each is at a different hospital or school with different characteristics. But if indeed there are differences between the source distributions, it is natural to learn different predictors $\mathbf{w}_i$ for each machine, so that $\mathbf{w}_i$ is good for the distribution typical to that machine. In this regard, our distributed multi-task learning problem is more similar to single-source problems, in that machines could potentially learn on their own given enough samples and enough time. Furthermore, availability of other machines just makes the problem easier by allowing transfer between the machine, thus reducing the sample complexity and runtime. The goal, then, is to leverage as much transfer as possible, while limiting communication and runtime. As with single-source problems, we compare our method to the two baselines, where we would like to be much better than the "local" approach, achieving performance nearly as good as the "centralized" approach, but with minimal communication and efficient runtime.

Related Work To the best of our knowledge, the only previous discussion of distributed multi-task learning is Dinuzzo et al. [2011], which considered a different setting with an almost orthogonal goal: a client-server architecture, where the server collects data from different clients, and sends sufficient information that might be helpful for each client to solve its own task. Their emphasis is on preserving privacy, but their architecture is communication-heavy as the entire data set is communicated to the central server, as in the "centralized" base line. On the other hand, we are mostly concerned with communication costs, but, for the time being, do not address privacy concerns.

In terms of distributed methods for uncovering shared sparsity, Baron et al. [2009] propose a “group orthogonal matching pursuit”-type algorithm. Their algorithm (DCS-SOMP) can be naturally implemented in a distributed way. However, (i) they consider only a noiseless setting, whereas we allow for noise; (ii) they do not establish any guarantees for DCS-SOMP, presenting only empirical results; and (iii) they require $\mathcal{O}(|S|)$ rounds of communication, with overall communication $\mathcal{O}(|S|p)$ per machine, whereas our DSML procedure requires only a single round and $\mathcal{O}(p)$ communication.

6.2 Problem set-up

We consider the following multi-task linear regression model with m tasks:

$$y_t = X_t\mathbf{w}_t^* + \epsilon_t, \qquad t = 1, \ldots, m, \tag{6.1}$$

where $X_t \in \mathbb{R}^{n_t \times p}$, $y_t \in \mathbb{R}^{n_t}$, and $\epsilon_t \sim N(0, \sigma_t^2 I) \in \mathbb{R}^{n_t}$ is a noise vector, and $\mathbf{w}_t^*$ is the unknown vector of coefficients for the task t. For notation simplicity we assume each task has equal sample size and the same noise level, that is, we assume, $n_1 = n_2 = \ldots = n$ and $\sigma_1 = \sigma_2 = \ldots = \sigma$. We will be working in a high-dimensional regime with p possibly larger than n, however, we will assume that each $\mathbf{w}_t^*$ is sparse, that is, few components of $\mathbf{w}_t^*$ are different from zero. Furthermore, we assume that the support between the tasks is shared. In particular, that support$(\mathbf{w}_t^*) = \{j \in [p] : \mathbf{w}_{tj} \neq 0\} \subset S$, with $s = |S| \ll n$. Suppose the data sets $(X_1, y_1), \ldots, (X_m, y_m)$ are distributed across machines, our goal is to estimate $\{\mathbf{w}_t^*\}_{t=1}^m$ as accurately as possible, while maintaining low communication cost.

The lasso estimate for each task t is given by:

$$\widehat{\mathbf{w}}_t = \arg\min_{\mathbf{w}_t} \frac{1}{n}||y_t - X_t\mathbf{w}_t||_2^2 + \lambda_t||\mathbf{w}_t||_1. \tag{6.2}$$

The multi-task estimates are given by the joint optimization:

$$\{\widehat{\mathbf{w}}_t\}_{t=1}^m = \arg\min_{\{\mathbf{w}_t\}_{t=1}^m} \frac{1}{mn}\sum_{t=1} ||y_t - X_t\mathbf{w}_t||_2^2 + \lambda \text{pen}(\{\mathbf{w}_t\}_{t=1}^m), \tag{6.3}$$

where $\text{pen}(\{\mathbf{w}_t\}_{t=1}^m)$ is the regularizaton that promote group sparse solutions. For example, the group lasso penalty uses $\text{pen}(\{\mathbf{w}_t\}_{t=1}^m) = \sum_{j\in[p]}\sqrt{\sum_{t\in m}\mathbf{w}_{tj}^2}$ Yuan and Lin [2006], while the iCAP uses $\text{pen}(\{\mathbf{w}_t\}_{t=1}^m) = \sum_{j\in[p]}\max_{t=1,\ldots,m}|\mathbf{w}_{tj}|$ Zhao et al. [2009], Liu et al. [2009]. In a distributed setting, one could potentially minimize (6.3) using a distributed consensus procedure (see Section 6.1.1), but such an approach would generally require multiple rounds of communication. Our procedure, described in the next section, lies in between the local lasso (6.2) and centralized estimate (6.3), requiring only one round of communication to compute, while still ensuring much of the statistical benefits of using group regularization.

6.3 Methodology

In this section, we detail our procedure for performing estimation under model in (6.1). Algorithm 16 provides an outline of the steps executed by the worker nodes and the master node, which are explained in details below.

Algorithm 16 DSML:Distributed debiased Sparse Multi-task Lasso.

```
Workers:
for $t = 1, 2, \ldots, m$ do
    Each worker obtains $\widehat{\mathbf{w}}_t$ as a solution to a local lasso in (6.2) Each worker obtains $\widehat{\mathbf{w}}_t^u$ the debiased lasso estimate in (6.4) and sends it to the master if Receive $\widehat{S}(\Lambda)$ from the master then
        Calculate final estimate $\widetilde{\mathbf{w}}_t$ in (6.6).
    end
end
Master:
if Receive $\{\widehat{\mathbf{w}}_t^u\}_{t=1}^m$ from all workers then
    Compute $\widehat{S}(\Lambda)$ by group hard thresholding in(6.5) and send the result back to every worker.
end
```

Recall that each worker node contains data for one task. That is, a node t contains data (X_t, y_t). In the first step, each worker node solves a lasso problem locally, that is, a node t minimizes the program in (6.2) and obtains $\widehat{\mathbf{w}}_t$. Next, a worker node constructs a debiased lasso estimator $\widehat{\mathbf{w}}_t^u$ by performing one Newton step update on the loss function, starting at the estimated value $\widehat{\mathbf{w}}_t$:

$$\widehat{\mathbf{w}}_t^u = \widehat{\mathbf{w}}_t + n^{-1} M_t X_t^T (y_t - X_t \widehat{\mathbf{w}}_t), \tag{6.4}$$

where $n^{-1} X_t^T (y_t - X_t \widehat{\mathbf{w}}_t)$ is a subgradient of the loss and the matrix $M_t \in \mathbb{R}^{p \times p}$ serves as an approximate inverse of the Hessian. The idea of debiasing the lasso estimator was introduced in the recent literature on statistical inference in high-dimensions Zhang and Zhang [2013], Van de Geer et al. [2014], Javanmard and Montanari [2014]. By removing the bias introduced through the ℓ_1 penalty, one can estimate the sampling distribution of a component of $\widehat{\mathbf{w}}_t^u$ and make inference about the unknown parameter of interest. In this chapter, we will also utilize the sampling distribution of the debiased estimator, however, with a different goal in mind. The above mentioned papers proposed different techniques to construct the matrix M. Here, we adopt the approach proposed in Javanmard and Montanari [2014], as it leads to the weakest assumption on the model in (6.1): each machine uses a matrix $M_t = (\widehat{m}_{tj})_{j=1}^p$ with rows:

$$\begin{aligned} \widehat{m}_{tj} = \arg \min_{m_j \in \mathbb{R}^p} \quad & m_j^T \widehat{\Sigma}_t m_j \\ \text{s.t.} \quad & ||\widehat{\Sigma}_t m_j - e_j||_\infty \le \mu. \end{aligned}$$

where e_j is the vector with j-th component equal to 1 and 0 otherwise and $\widehat{\Sigma}_t = n^{-1} X_t^T X_t$.

After each worker obtains the debiased estimator $\widehat{\mathbf{w}}_t^u$, it sends it to the central machine. After debiasing, the estimator is no longer sparse and as a result each worker communicates p numbers to the master node. It is at the master where shared sparsity between the task coefficients gets utilized. The master node concatenates the received estimators into a matrix $\widehat{B} = (\widehat{\mathbf{w}}_1^u, \widehat{\mathbf{w}}_2^u, ..., \widehat{\mathbf{w}}_m^u)$. Let $\widehat{B}_j$ be the j-th row of $\widehat{B}$. The master performs the hard group thresholding to obtain an estimate of S as

$$\widehat{S}(\Lambda) = \{j \mid ||\widehat{B}_j||_2 > \Lambda\}. \tag{6.5}$$

The estimated support $\widehat{S}(\Lambda)$ is communicated back to each worker, which then use the estimate of the support to filter their local estimate. In particular, each worker produces the final estimate:

$$\widetilde{\mathbf{w}}_{tj} = \begin{cases} \widehat{\mathbf{w}}_{tj}^u & \text{if } j \in \widehat{S}(\Lambda) \\ 0 & \text{otherwise.} \end{cases} \tag{6.6}$$

Extension to multitask classification. DSML can be generalized to estimate multi-task generalized linear models. We be briefly outline how to extend DSML to a multi-task logistic regression model, where $y_{tk} \in \{-1, 1\}$ and $\forall k = 1, \ldots, n, t = 1, \ldots, m$:

$$P(y_{tk}|X_{tk}) = \frac{\exp\left(\frac{1}{2}y_{tk}X_{tk}\mathbf{w}_t^*\right)}{\exp\left(-\frac{1}{2}y_{tk}X_{tk}\mathbf{w}_t^*\right) + \exp\left(\frac{1}{2}y_{tk}X_{tk}\mathbf{w}_t^*\right)}. \tag{6.7}$$

First, each worker solves the ℓ_1-regularized logistic regression problem

$$\widehat{\mathbf{w}}_t = \arg\min_{\mathbf{w}_t} \frac{1}{n} \sum_{k \in [n]} \log(1 + \exp(-y_{tk}X_{tk}\mathbf{w}_t)) + \lambda_t ||\mathbf{w}_t||_1.$$

Let $W_t \in \mathbb{R}^{n \times n}$ be a diagonal weighting matrix, with a k-th diagonal element

$$W_{t(kk)} = \frac{1}{1 + \exp(-X_{tk}\widehat{\mathbf{w}}_t)} \cdot \frac{\exp(-X_{tk}\widehat{\mathbf{w}}_t)}{1 + \exp(-X_{tk}\widehat{\mathbf{w}}_t)},$$

which will be used to approximately invert the Hessian matrix of the logistic loss. The matrix $M_t = (\widehat{m}_{tj})_{j=1}^p$, which serves as an approximate inverse of the Hessian, in the case of logistic regression can be obtained as a solution to the following optimization problem:

$$\begin{aligned}\widehat{m}_{tj} = \arg\min_{m_{tj}\in\mathbb{R}^p} \quad & m_{tj}^T {X_t}^T W_t X_t m_{tj} \\ \text{s.t.} \quad & \|n^{-1} X_t^T W_t X_t m_{tj} - e_j\|_\infty \leq \mu.\end{aligned}$$

Finally, the debiased estimator is obtained as

$$\widehat{\mathbf{w}}_t^u = \widehat{\mathbf{w}}_t + \frac{1}{n} M_t {X_t}^T \left(\frac{1}{2}(y_t + 1) - (1 + \exp(-X_t \widehat{\mathbf{w}}_t))^{-1} \right),$$

and then communicated to the master node. The rest of procedure is as described before.

6.4 Theoretical Analysis

In this section, we present our main theoretical results for the DSML procedure described in the previous section. We start by describing assumptions that we make on the model in (6.1). We analyze here the well-specified random model, i.e. when samples are generated by the model (6.1), with rows of X_t drawn from some sub-Gaussian distribution (possibly a different distribution for each task). In the Appendix we also analyze the "fixed-design" setting, i.e. in terms of properties of the sample matrices X_t.

We assume rows of X_t are drawn from a subgaussian distribution with covariance matrix $\mathbb{E}[n^{-1} X_t^T X_t] = \Sigma_t$.

We assume the distribution of rows of X_t for each task have bounded subgaussian norm $\max_t \max_k \|X_{tk}\|_{\psi_2} \leq \sigma_X$ Vershynin [2010]. Let Σ_t be the covariance for task t. We rely on upper and lower bounds on the eigenvalues of the covariances Σ_t of the data for each task t: $\forall_t \lambda_{\min} I \preceq \Sigma_t \preceq \lambda_{\max} I$. Our analysis is based on assuming $\lambda_{\min} > 0$. We also rely on a bound on the elements of precision matrices, $\forall_t \left|\Sigma_t^{-1}\right|_\infty \leq K$. We can always take $K = 1/\lambda_{\min}$, but we often have a much tighter bound.

The following theorem is our main result, which is proved in appendix.

Theorem 53. *With the assumptions and notation above, if λ in (6.2) is chosen as $\lambda_t = 4\sigma\sqrt{\frac{\log p}{n}}$, and the coefficients in (6.1) satisfy*

$$\min_{j \in S} \sqrt{\sum_{t \in [m]} (\beta_{tj}^*)^2} \geq 6K\sigma\sqrt{\frac{m + \log p}{n}} + \frac{C\sigma_X^4 \lambda_{\max}^{1/2} \sigma |S| \sqrt{m} \log p}{\lambda_{\min}^{3/2} n} := 2\Lambda^*, \tag{6.8}$$

where $C < 5000$ is some numeric constant, then the support estimated by the master node satisfies $\widehat{S}(\Lambda^*) = S$ with probability at least $1 - mp^{-1}$.

Based on Theorem 53, we have the following corollary that characterizes the parameter and prediction errors of DSML, with the proof given in the appendix:

Corollary 54. *With the same choice of λ_t, with probability at least $1 - mp^{-1}$, we have*

$$\sum_{j=1}^{p} ||\widetilde{B}_j - B_j||_2 \leq 6K|S|\sigma\sqrt{\frac{m + \log p}{n}} + \frac{C\sigma_X^4 \lambda_{\max}^{1/2} \sigma |S|^2 \sqrt{m} \log p}{\lambda_{\min}^{3/2} n},$$

$$\frac{\sum_{t=1}^{m} (\mathbb{E}_{X_t}(X_t\widetilde{\beta}_t - X_t\beta_t^*))^2}{nm} \leq \frac{36K^2|S|\sigma^2}{n}\left(1 + \frac{\log p}{m}\right) + \frac{C^2\sigma_X^8 \lambda_{\max} \sigma^2 |S|^3 (\log p)^2}{\lambda_{\min}^3 n^2}.$$

Let us compare these guarantees to the group lasso. For DSML Corollary 2 yields:

$$\frac{1}{\sqrt{m}} \sum_{j=1}^{p} ||\widetilde{B}_j - B_j||_2 \lesssim \frac{|S|}{\sqrt{n}}\sqrt{1 + \frac{\log p}{m}} + \frac{|S|^2 \log p}{n}, \tag{6.9}$$

where $a(n) \gtrsim b(n)$ means that for some c, N, $a(n) > c \cdot b(n), \forall n > N$. When using the group lasso, the restricted eigenvalue condition is sufficient for obtaining error bounds and

Approach	Min signal strength	Strength type
Lasso	$\sqrt{\frac{\log p}{n}}$	Element-wise
Group lasso	$\sqrt{\frac{1}{n}\left(1+\frac{\log p}{m}\right)}$	Row-wise
DSML	$\sqrt{\frac{1}{n}\left(1+\frac{\log p}{m}\right)+\frac{\lvert S\rvert\log p}{n}}$	Row-wise

Table 6.2: Lower bound on coefficients required to ensure support recovery with p variables, m tasks, n samples per task and a true support of size $|S|$.

following holds for the group lasso [Corollary 4.1 of Lounici et al., 2011]:

$$\frac{1}{\sqrt{m}}\sum_{j=1}^{p}||\widetilde{B}_j - B_j||_2 \leq \frac{32\sqrt{2}\sigma|S|}{\kappa\sqrt{n}}\sqrt{\left(1+\frac{2.5\log p}{m}\right)} \lesssim \frac{|S|}{\sqrt{n}}\sqrt{1+\frac{\log p}{m}}, \tag{6.10}$$

which is min-max optimal (up to a logarithmic factor). DSML matches this bound when $n \gtrsim \frac{m|S|^2(\log p)^2}{(m+\log p)}$. Similarly for prediction DSML attains:

$$\frac{1}{nm}\sum_{t=1}^{m}(\mathbb{E}_{X_t}(X_t\widetilde{\beta}_t - X_t\beta_t^*))^2 \lesssim \frac{|S|\sigma^2}{n}\left(1+\frac{\log p}{m}\right) + \frac{\sigma^2|S|^3(\log p)^2}{n^2}, \tag{6.11}$$

which in the same regime matches the group lasso minimax optimal rate:

$$\frac{1}{nm}\sum_{t=1}^{m}(\mathbb{E}_{X_t}(X_t\widetilde{\beta}_t - X_t\beta_t^*))^2 \leq \frac{128|S|\sigma^2}{\kappa n}\left(1+\frac{2.5\log p}{m}\right) \lesssim \frac{|S|\sigma^2}{n}\left(1+\frac{\log p}{m}\right). \tag{6.12}$$

In both cases, as long as m is not too large, we have a linear improvement over Lasso, which corresponds to (6.10) and (6.12) with $m = 1$.

The discussion of support recovery is more complex as typically more stringent conditions (for example, mutual incoherence or irrepresentable condition) are imposed on X_t for lasso and group lasso to achieve sparsistency. See Van De Geer et al. [2009] for an extensive discussion of different conditions used in the literature. In any case, we can also compare the minimal signal strength required for DSML to that required by lasso and group lasso.

Let $B = [\beta_1, \beta_2, \dots, \beta_m] \in \mathbb{R}^{p \times m}$ be the matrix of true coefficients. Simplifying (6.8), we have that our procedure requires the minimum signal strength to satisfy

$$\min_{j \in S} \frac{1}{\sqrt{m}} ||B_j||_2 \gtrsim \sqrt{\frac{1}{n}\left(1 + \frac{\log p}{m}\right)} + \frac{|S| \log p}{n}. \tag{6.13}$$

For the centralized group lasso, the standard analysis assumes a stronger condition on the data, namely that X_t satisfies mutual incoherence with parameter α and sparse eigenvalue condition [Lounici et al., 2011] (see Van De Geer et al. [2009] for an extensive discussion of different conditions used in the literature to guarantee support recovery). Under this condition, group lasso recovers the support if [Corollary 5.2 of Lounici et al., 2011]:

$$\min_{j \in S} \frac{1}{\sqrt{m}} ||B_j||_2 \geq \frac{4\sqrt{2} C_{\alpha,\kappa} \sigma}{\sqrt{n}} \sqrt{1 + \frac{2.5 \log p}{m}} \gtrsim \sqrt{\frac{1}{n}\left(1 + \frac{\log p}{m}\right)}. \tag{6.14}$$

where $C_{\alpha,\kappa}$ depends only on the mutual incoherence and sparse eigenvalue of X_t. Under the irrepresentable condition on X_t, which is weaker than the mutual incoherence [Van De Geer et al., 2009], the lasso requires the signal to satisfy [Bunea et al., 2008, Wainwright, 2009]:

$$\min_{t \in [m]} \min_{j \in S} |\beta^*_{tj}| \geq C_{\gamma,\kappa} \sigma \sqrt{\frac{\log p}{n}} \gtrsim \sqrt{\frac{\log p}{n}} \tag{6.15}$$

for some $C_{\gamma,\kappa}$, which depends only on the irrepresentable condition and the sparse eigenvalue. Ignoring for the moment the differences in the conditions on the design matrix, there are two advantages of the multitask group lasso over the local lasso: (1) relaxing the signal strength requirement to a requirement on the *average* strength across tasks (i.e. any single coefficient can be arbitrarily small, even zero); and (2) a reduction by a factor of m on the $\log p$ term. Similarly to the group lasso, DSML requires a lower bound only on the average signal strength, not on any individual coefficient. And as long as $m \ll n$, or more precisely $n \gtrsim \frac{m|S|^2 (\log p)^2}{\kappa^2 (m + \log p)}$, DSML enjoys the same linear reduction in the dominant term of

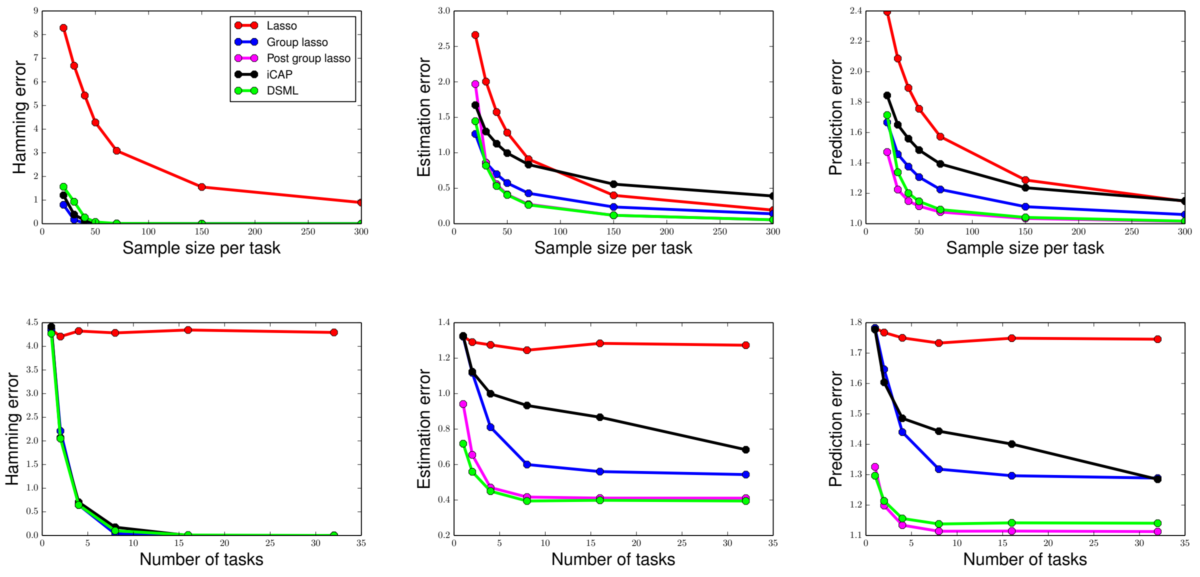


Figure 6.1: Hamming distance, estimation error, and prediction error for multi-task regression with $p = 200$. Top row: the number of tasks $m = 10$. Sample size per tasks is varied. Bottom row: Sample size $n = 50$. Number of tasks m varied.

the required signal strength, matching the leading term of the group lasso bound. This is summarized in Table 6.2.

To better elucidate the differences in conditions, in the Appendix, we carry out an analysis of DSML in a "fixed design" setting. We show that Theorem 53 and Corollary 54 for fixed X_t, as long two conditions hold: generalized coherence (a weakening of the mutual incoherence condition) and restricted eigenvalue. These conditions are stronger than what is required for only small parameter and prediction error using the Lasso and Group Lasso (restricted eigenvalue on its own is sufficient, not need for generalized coherence), but similar and in a sense weaker than what is required for support recovery. See the Appendix for precise definitions and statements of the results.

It might be interesting to ask why debiasing is needed, and simple "local lasso" plus "group thresholding" will not work. To see this, let us consider a simple case of "weak signals group": suppose there is a coordinate in the support S where signals for all models are weak, on the order of $\mathcal{O}\left(\sqrt{\frac{1}{n}\left(1+\frac{\log p}{m}\right)}+\frac{|S|\log p}{n}\right)$. The local lasso cannot distinguish this weak signal from zeros, making it always remove the coordinate from the support, if the estimator wants to ensure removing all zero coordinates from the estimated support. In contrast, DSML will save this weak signal, making the estimated support consistent. A concrete example illustrating the situation will be presented in the experiments section.

6.4.1 Fixed design analysis

In this section, we present our theoretical results for the DSML procedure for fixed design, we will state the results without proof since the process is essentially the same as the case for random design. The results and comparisons are summarized in Table 6.3 and 6.4. We start by describing assumptions that we make on the model in (6.1). We assume the following condition on the design matrices $\{X_t\}_{t=1}^m$.

Approach	Communication	Assumptions	Min signal strength	Strength type
Lasso	0	Mutual Incoherence Sparse Eigenvalue	$\sqrt{\frac{\log p}{n}}$	Element-wise
Group lasso	$\mathcal{O}(np)$	Mutual Incoherence Sparse Eigenvalue	$\sqrt{\frac{1}{n}\left(1+\frac{\log p}{m}\right)}$	Row-wise
DSML	$\mathcal{O}(p)$	Generalized Coherence Restricted Eigenvalue	$\sqrt{\frac{1}{n}\left(1+\frac{\log p}{m}\right)}+\frac{\|S\|\log p}{n}$	Row-wise

Table 6.3: Conditions on the design matrix, and corresponding lower bound on coefficients required to ensure support recovery with p variables, m tasks, n samples per task and a true support of size $|S|$.

A1 (Restricted Eigenvalues): Let $\mathcal{C}(s,L)=\{\Delta\in\mathbb{R}^p \mid ||\Delta_{U^c}||_1\le L||\Delta_U||_1, U\subseteq [p], |U|\le s\}$. There exists a constant $\kappa>0$ such that

$$\min_{\Delta\in\mathcal{C}(|S|,L)} \Delta^T\widehat{\Sigma}_t\Delta \ge \kappa||\Delta_U||_2^2, \qquad t=1,\dots,m.$$

Approach	Assumptions	ℓ_1/ℓ_2 estimation error	Prediction error
Lasso	Restricted Eigenvalue	$\sqrt{\frac{\lvert S\rvert^2 \log p}{n}}$	$\frac{\lvert S\rvert \log p}{n}$
Group lasso	Restricted Eigenvalue	$\frac{\lvert S\rvert}{\sqrt{n}}\sqrt{1+\frac{\log p}{m}}$	$\frac{\lvert S\rvert}{n}\left(1+\frac{\log p}{m}\right)$
DSML	Generalized Coherence Restricted Eigenvalue	$\frac{\lvert S\rvert}{\sqrt{n}}\sqrt{1+\frac{\log p}{m}}+\frac{\lvert S\rvert^2\log p}{n}$	$\frac{\lvert S\rvert}{n}\left(1+\frac{\log p}{m}\right)+\frac{\lvert S\rvert^3(\log p)^2}{n^2}$

Table 6.4: Conditions on the design matrix, and comparison of parameter estimation errors and prediction errors. The DSML guarantees improve over Lasso and have the same leading term as the Group lasso as long as $m < n/(|S|^2 \log p)$.

There exists a constant $\phi_{\max} < \infty$ such that

$$\max_{\Delta \in \mathbb{R}^p} \Delta_S^T \widehat{\Sigma}_t \Delta_S < \phi_{\max} ||\Delta_S||_2^2.$$

The above assumption is commonly assumed in the literature in order to establish consistent estimation in high-dimensions Bickel et al. [2009]. **A1** imposes restrictions directly on the sample covariances $\widehat{\Sigma}_t$, however, it is well known that the assumption will hold with high-probability when rows of X_t are i.i.d. sub-gaussian or sub-exponential random vectors with population covariance satisfying **A1** Raskutti et al. [2010], Rudelson and Zhou [2013].

We will also need the following notion of coherence of the design matrices.

Definition 55 (Generalized Coherence)**.** *For matrices $X \in \mathbb{R}^{n\times p}$ and $M = (m_1, \ldots, m_p) \in \mathbb{R}^{p\times p}$, let*

$$\mu(X, M) = \max_{j\in[p]} ||\Sigma m_j - e_j||_\infty$$

be the generalized coherence parameter between X and M, where $\Sigma = n^{-1}X^T X$. Furthermore, let $\mu^* = \min_{t\in[m]} \min_{M\in\mathbb{R}^{p\times p}} \mu(X_t, M)$ be the minimum generalized coherence.

This assumption is more relaxed than the mutual coherence parameter Donoho and Huo [2001]. As shown in Theorem 2.4 of Javanmard and Montanari [2014], $\mu(X_t, \Sigma^{-1}) \leq 2\sqrt{\log(p)/n}$ with high-probability when the rows of X_t are i.i.d. sub-gaussian vectors with covariance matrix Σ.

The following theorem is our main result, which is proved in appendix.

Theorem 56. *Assume that* **A1** *holds and that the generalized coherence condition satisfies* $\mu^* \leq C_\mu\sqrt{\frac{\log p}{n}}$ *for some constant* C_μ. *Suppose* λ *in* (6.2) *was chosen as* $\lambda_t = A\sigma\sqrt{\frac{\log p}{n}}$ *with constant* $A > \sqrt{2}$. *Furthermore, suppose that the multi-task coefficients in* (6.1) *satisfy the following bound on the signal strength*

$$\min_{j\in S}\sqrt{\sum_{t\in[m]}(\mathbf{w}^*_{tj})^2} \geq \frac{2\sigma}{\sqrt{n}}\sqrt{\frac{512A^2C_\mu^2 m|S|^2(\log p)^2}{\kappa^2 n} + 6C_M^2 m + 2\sqrt{2}C_M^2\log p} := 2\Lambda^*, \tag{6.16}$$

where C_M *is a constant that only depends on* $\{M_t\}_{t=1}^m$. *Then the support estimated by the master node satisfies* $\widehat{S}(\Lambda^*) = S$ *with probability at least* $1 - mp^{1-A^2/2} - p^{-1}$.

Based on Theorem 53, we have the following corollary that characterizes estimation error and prediction risk of DSML, with the proof given in the appendix.

Corollary 57. *Suppose the conditions of Theorem 56 hold. With probability at least* $1 - mp^{1-A^2/2} - p^{-1}$, *we have*

$$\sum_{j=1}^p ||\widetilde{B}_j - B_j||_2 \leq \frac{|S|\sigma}{\sqrt{n}}\sqrt{\frac{512A^2C_\mu^2|S|^2(\log p)^2}{\kappa^2 n} + 6C_M^2 m + 2\sqrt{2}C_M^2\log p}$$

and

$$\frac{1}{nm}\sum_{t=1}^m ||X_t(\widetilde{\mathbf{w}}_t - \mathbf{w}_t^*)||_2^2 \leq \frac{\phi_{\max}|S|\sigma^2}{n}\left(\frac{512A^2C_\mu^2 m|S|^2(\log p)^2}{\kappa^2 n} + 6C_M^2 + \frac{2\sqrt{2}C_M^2\log p}{m}\right).$$

6.5 Experiments

Our first set of experiments is on simulated data. We generated synthetic data according to the model in (6.1) and in (6.7). Rows of X_t are sampled from a mean zero multivariate

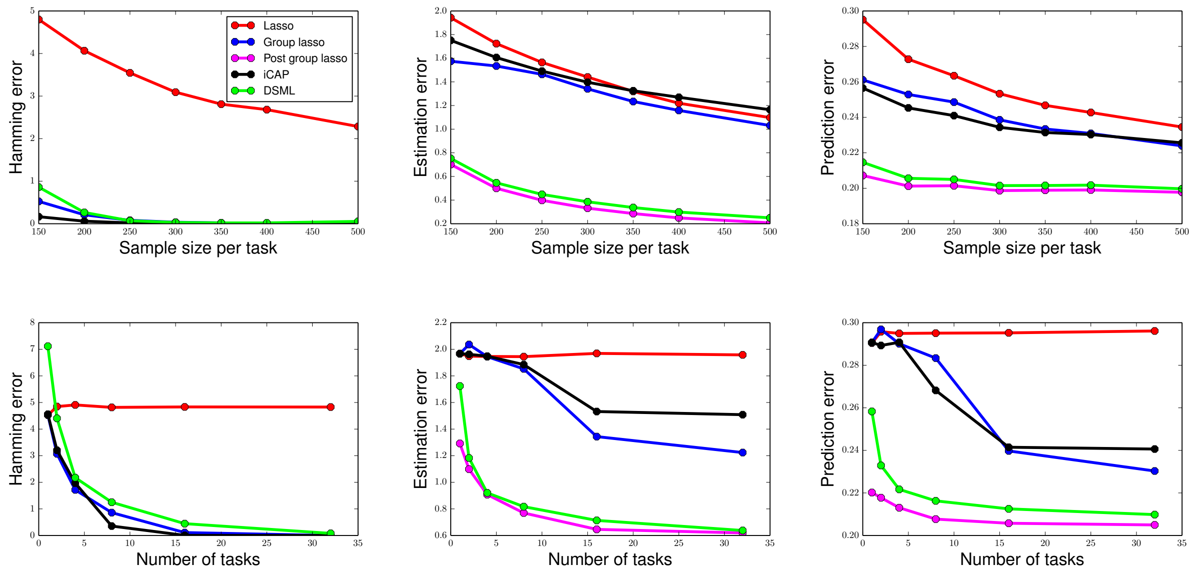


Figure 6.2: Hamming distance, estimation error, and prediction error for multi-task classification with $p = 200$. Top row: the number of tasks $m = 10$. Sample size per tasks is varied. Bottom row: Sample size $n = 150$. Number of tasks m varied.

normal with the covariance matrix $\Sigma = (\Sigma_{ab})_{a,b\in[p]}$, $\Sigma_{ab} = 2^{-|a-b|}$. The data dimension p is set to 200, while the number of true relevant variables s is set to 10. Non-zero coefficients of β are generated uniformly from $[0, 1]$. Variance σ^2 is set to 1. Our simulation results are averaged over 200 independent runs.

We investigate how performance of various procedures changes as a function of problem parameters (n, p, m, s). We compare the following procedures: i) local lasso, ii) group lasso, iii) refitted group lasso, where a worker node performs ordinary least squares on the selected support, iv) iCAP, and v) DSML. The parameters for local lasso, group lasso and iCAP were tuned to achieve the minimal Hamming error in variable selection. For DSML, to debias the output of local lasso estimator, we use $\mu = \sqrt{\log p/n}$. The thresholding parameter Λ is also optimized to achive the best variable selection performance. The simulation results for regression are shown in Figure 7.3. In terms of support recovery (measured by Hamming distance), Group lasso, iCAP, and DSML all perform similarly and significantly better than the local lasso. In terms of estimation error, lasso perform the worst, while DSML and refitted group lasso perform the best. This might be a result of bias removal introduced by regularization. Since the group lasso recovers the true support in most cases, refitting on it yields the maximum likelihood estimator on the true support. It is remarkable that DSML performs almost as well as this oracle estimator.

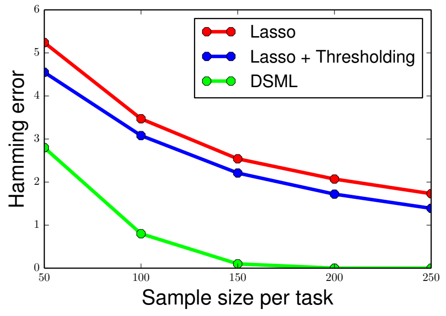


Figure 6.3: Comparison of DSML with “Local lasso + thresholding” on a synthetic example.

Figure 7.4 shows the simulation results for classification. Similar with the regression case, we make the following observations: i) The group sparsity based approaches, including DSML, significantly outperform the individual lasso; ii) In terms of Hamming variable selection error, DSML performs slightly worse than group lasso and iCAP. While in terms of estimation error and prediction error, DSML performs much better than group lasso and icap. Given the fact that group lasso recovers the true support in most cases, refitted group lasso is equivalent to oracle maximum likelihood estimator. It is remarkable that DSML only performs slightly worse than refitted group lasso; iii) The advantage of DSML, as well as group lasso over individual lasso, becomes more and more significant with the increase in number of tasks.

To illustrate why debasing is necessary, and a naive “local lasso + centralized thresholding” approach will not work, we also performed a simple simulation with the following setup: we divide support set S into a strong signal group $S_s \subset S$ and a weak signal group $S_w \subset S$, with the coefficients of β in S_s generated uniformly from $[0, 1]$, while the ones in S_w

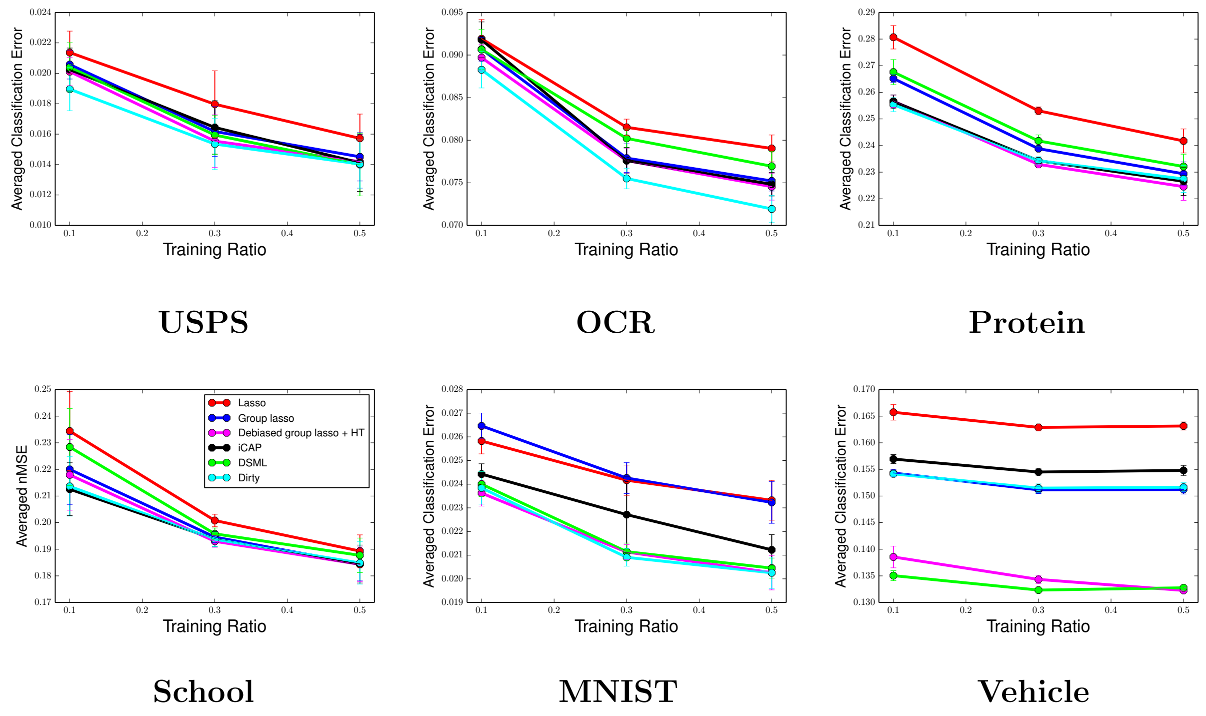


Figure 6.4: Comparison on real world datasets.

generated uniformly from $[0, 0.4]$. We test this setting on a multi-task regression problem with $p = 100$, the hamming selection error was shown in Figure 6.3, selecting the best regularization and thresholding parameter. We can see that the "Local lasso + thresholding" approach only works slightly better than lasso, while DSML improved significantly on both.

We have also evaluated DSML on the following real world data sets:

School. This is a widely used dataset for multi-task learning [Argyriou et al., 2008]. The goal is to predict the students' performance at London's secondary schools. There are 27 attributes for each student. The tasks are naturally divided according to different schools. We only considered schools with at least 200 students, which results in 11 tasks.

Protein. The task is to predict the protein secondary structure [Sander and Schneider, 1991]. We considered three binary classification tasks here: coil vs helix, helix vs strand, strand vs coil. The dataset consists of 24,387 instances in total, each with 357 features.

OCR. We consider the optical character recognition problem. Data were gathered by

Rob Kassel at the MIT Spoken Language Systems Group [1]. Following [Obozinski et al., 2010], we consider the following 9 binary classification task: c vs e, g vs y, g vs s, m vs n, a vs g, i vs j, a vs o, f vs t, h vs n. Each image is represented by 8×16 binary pixels.

MNIST. This is a handwritten digit recognition dataset [2]. Each image is represented by 784 pixels. We considered the following 5 binary classification task: 2 vs 4, 0 vs 9, 3 vs 5, 1 vs 7, 6 vs 8.

USPS. This dataset consists handwritten images from envelopes by the U.S. Postal Service. We considere the following 5 binary classification task: 2 vs 4, 0 vs 9, 3 vs 5, 1 vs 7, 6 vs 8. Each image is represented by 256 pixels.

Vehicle. We considered the vehicle classification problem in distributed sensor networks [Duarte and Hu, 2004] with 3 binary classification task: AAV vs DW, AAV vs noise, DW vs noise. There are 98,528 instances in total, each instances is described by 50 acoustic features and 50 seismic features.

In addition to the procedures used in the previous section, we also compare against the dirty model Jalali et al. [2010], as well as the centralized approach that first debias the group lasso, then perform group hard thresholding. Tuning parameters for these procedures were chosen based on performance on held-out data set. All regularization or thresholding parameters were tuned to be optimal using a 20% held-out validation dataset. We vary the training sample size as 10%, 30% and, 50% of the total data set size and report the performance on the test set (normalized Mean Squared Error for regression and classification error for classification). Figure 7.2 shows the results. We have the following general observations. Local lasso performs the worst, which demonstrates that utilizing group sparsity helps to improve the prediction performance. Our DSML methods performs comparably with to the state-of-the-art centralized approaches. Debiasing group lasso followed by hard thresholding

1. `http://www.seas.upenn.edu/~taskar/ocr/`

2. `http://yann.lecun.com/exdb/mnist/`

compares favorably to group lasso and has similar performance to dirty model.

6.6 Proofs of technical results

6.6.1 Proof of Theorem 53

We first introduce the following lemma.

Lemma 58. *When the rows of $X_1, \ldots, X_t$ are independent subgaussian random vectors, with mean zero, covariance $\Sigma_1, ..., \Sigma_t$, respectively. Let*

$$C_M = \max_{t\in[m]} \max_{j\in[p]} \left(M_t^T \left(\frac{X_t^T X_t}{n} \right) M_t \right)_{jj}.$$

Then with probability at least $1 - 2mp \exp(-cn) - 2mp^{-2}$ for some constant c, we have

$$C_M \le 2 \max_{t\in[m]} \max_{j\in[p]} (\Sigma_t^{-1})_{jj}.$$

Proof. As shown in Theorem 2.4 of Javanmard and Montanari [2014], Σ_t^{-1} will be a feasible solution for the problem of estimating M_t. Since we're minimizing $(M_t^T \widehat{\Sigma}_t M_t)_{jj}$, we must have

$$\max_{j\in[p]} (M_t^T \widehat{\Sigma}_t M_t)_{jj} \le \max_{j\in[p]} (\Sigma_t^{-1} \widehat{\Sigma}_t \Sigma_t^{-1})_{jj}.$$

Based on the concentration results of sub-exponential random variable Vershynin [2010], also Lemma 3.3 of Lee et al. [2017b], we know with probability at least $1 - 2p \exp(-cn)$ for some constant c, we have

$$\max_{j\in[p]} (\Sigma_t^{-1} \widehat{\Sigma}_t \Sigma_t^{-1})_{jj} \le 2 \max_{j\in[p]} (\Sigma_t^{-1})_{jj}.$$

Take an union bound over $t \in [m]$, we obtain with probability at least $1 - 2mp \exp(-cn)$,

$$C_M \le \max_{t\in[m]} \max_{j\in[p]} (M_t^T \widehat{\Sigma}_t M_t)_{jj} \le \max_{t\in[m]} \max_{j\in[p]} (\Sigma_t^{-1} \widehat{\Sigma}_t \Sigma_t^{-1})_{jj} \le 2 \max_{t\in[m]} \max_{j\in[p]} (\Sigma_t^{-1})_{jj}.$$

$\square$

Now we are ready to prove Theorem 1, recall the model assumption

$$y_t = X_t\beta_t^* + \epsilon_t, \qquad t = 1, \ldots, m, \tag{6.17}$$

and the debiased estimation

$$\widehat{\beta}_t^u = \widehat{\beta}_t + n^{-1}M_tX_t^T(y_t - X_t\widehat{\beta}_t), \tag{6.18}$$

we have

$$\begin{aligned}\widehat{\beta}_t^u =& \widehat{\beta}_t + \frac{1}{n}M_tX_t^T(X_t\beta_t^* - X_t\widehat{\beta}_t) + \frac{1}{n}M_tX_t^T\epsilon_t \\ =& \beta_t^* + (M_t\widehat{\Sigma}_t - I)(\beta_t^* - \widehat{\beta}_t) + \frac{1}{n}M_tX_t^T\epsilon_t.\end{aligned}$$

For the term $(M_t\widehat{\Sigma}_t - I)(\beta_t^* - \widehat{\beta}_t)$, define

$$C_\mu = 10e\sigma_X^4\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}},$$

we have the following bound

$$\begin{aligned}||(M_t\widehat{\Sigma}_t - I)(\beta_t^* - \widehat{\beta}_t)||_\infty \leq& \max_j ||\widehat{\Sigma}_t m_{tj} - e_j||_\infty ||\beta_t^* - \widehat{\beta}_t||_1 \\ \leq_P& C_\mu\sqrt{\frac{\log p}{n}} \cdot \frac{16A}{\kappa}\sigma|S|\sqrt{\frac{\log p}{n}} \\ =& \frac{16AC_\mu\sigma|S|\log p}{\kappa n}.\end{aligned} \tag{6.19}$$

Noticed that

$$n^{-1}M_tX_t^T\epsilon_t \sim \mathcal{N}\left(0, \frac{\sigma^2M_t\widehat{\Sigma}_tM_t{}^T}{n}\right).$$

Our next step uses a result on the concentration of χ^2 random variables. For any coordinate

j, we have

$$\sum_{i=1}^{m} \left(n^{-1} e_j^T M_t X^T \epsilon_t\right)^2 \leq \frac{C_M^2 \sigma^2}{n} \sum_{i=1}^{m} \xi_i^2,$$

where $(\xi_i)_{i\in[m]}$ are standard normal random variables. Using Lemma 61 with a weight vector

$$v = \left(\frac{C_M^2\sigma^2}{n}, \frac{C_M^2\sigma^2}{n}, \ldots, \frac{C_M^2\sigma^2}{n}\right)$$

and choosing $t = \sqrt{m} + \frac{\log p}{\sqrt{m}}$, we have

$$P\left\{ \frac{\left(\frac{C_M^2\sigma^2}{n}\right)\sum_{i=1}^m \xi_i^2}{\sqrt{2m}\left(\frac{C_M^2\sigma^2}{n}\right)} - \sqrt{\frac{m}{2}} > \sqrt{m} + \frac{\log p}{\sqrt{m}} \right\} \leq 2\exp\left(-\frac{\left(\sqrt{m} + \frac{\log p}{\sqrt{m}}\right)^2}{2 + 2\sqrt{2}(1 + \frac{\log p}{m})}\right).$$

A union bound over all $j \in [p]$ gives us that with probability at least $1 - p^{-1}$

$$\sum_{i\in[m]} \left(n^{-1} e_j^T M_t X^T \epsilon_t\right)^2 \leq 3m\left(\frac{C_M^2\sigma^2}{n}\right) + \sqrt{2}\log p\left(\frac{C_M^2\sigma^2}{n}\right), \qquad \forall j \in [p]. \tag{6.20}$$

Combining (6.19) and (6.20), we get the following estimation error bound:

$$\begin{aligned}
||\widehat{B}_j - B_j||_2 =& \sqrt{\sum_{i\in[m]} \left([M_t\widehat{\Sigma}_t - I)(\beta_t^* - \widehat{\beta}_t)]_j + \left[n^{-1} M_t X_t^T \epsilon_t\right]_j\right)^2} \\
\leq& \sqrt{\sum_{i\in[m]} 2\left([M_t\widehat{\Sigma}_t - I)(\beta_t^* - \widehat{\beta}_t)]_j^2 + \left[n^{-1} M_t X_t^T \epsilon_t\right]_j^2\right)} \\
\leq& \sqrt{\sum_{i\in[m]} \left(\frac{512A^2C_\mu^2\sigma^2|S|^2(\log p)^2}{\kappa^2 n^2}\right) + 6m\left(\frac{C_M^2\sigma^2}{n}\right) + 2\sqrt{2}\log p\left(\frac{C_M^2\sigma^2}{n}\right)} \\
=& \frac{\sigma}{\sqrt{n}}\sqrt{\frac{512A^2C_\mu^2 m|S|^2(\log p)^2}{\kappa^2 n} + 6C_M^2 m + 2\sqrt{2}C_M^2\log p} \\
\leq& \frac{91C_\mu\sigma|S|\sqrt{m}\log p}{\kappa n} + 3C_M\sigma\sqrt{\frac{m + \log p}{n}},
\end{aligned} \tag{6.21}$$

where the first inequality uses the fact $(a+b)^2 \le 2a^2 + 2b^2$, and the second inequality uses (6.19) and (6.20)), the last inequality uses the fact that $\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$. For every variable $j \notin S$, we have

$$||\widehat{B}_j||_2 \le \frac{91 C_\mu \sigma |S| \sqrt{m} \log p}{\kappa n} + 3 C_M \sigma \sqrt{\frac{m + \log p}{n}}.$$

plug in $\kappa \ge \frac{1}{2}\lambda_{\min}$, $C_\mu = 10e\sigma_X^4 \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}$, $C_M \le 2K$, we obtain

$$||\widehat{B}_j||_2 \le \frac{1820 e \sigma_X^4 \lambda_{\max}^{1/2} \sigma |S| \sqrt{m} \log p}{\lambda_{\min}^{3/2} n} + 6K\sigma \sqrt{\frac{m + \log p}{n}}.$$

From (6.21) and the choice of Λ^*, we see that all variables not in S will be excluded from $\widehat{S}$ as well. For every variable $j \in S$, we have

$$||\widehat{B}_j||_2 \ge ||B_j||_2 - ||\widetilde{B}_j - B_j||_2 \ge 2\Lambda^* - \Lambda^* = \Lambda^*.$$

Therefore, all variables in S will correctly stay in $\widehat{S}$ after the group hard thresholding.

6.6.2 Proof of Corollary 54

From Theorem 2 we have that $\widehat{S}(\Lambda^*) \subseteq S$ and

$$||\widetilde{B}_j - B_j||_2 \le \frac{1820 e \sigma_X^4 \lambda_{\max}^{1/2} \sigma |S| \sqrt{m} \log p}{\lambda_{\min}^{3/2} n} + 6K\sigma \sqrt{\frac{m + \log p}{n}}, \qquad (6.22)$$

with high probability. Summing over $j \in S$, we obtain the ℓ_1/ℓ_2 estimation error bound. For the prediction risk bound, we have

$$\begin{aligned}\frac{1}{nm}\sum_{t=1}^{m}||X_t(\widetilde{\beta}_t - \beta_t^*)||_2^2 \leq &\frac{\lambda_{\max}}{m}\sum_{i=1}^{m}||\widetilde{\beta}_t - \beta_t^*||_2^2\\ =&\frac{\lambda_{\max}}{m}\sum_{j=1}^{p}||\widetilde{B}_j - B_j||_2^2.\end{aligned}$$

Using (6.22) and the fact that $\widetilde{B} - B$ is row-wise $|S|$-sparse, we obtain the prediction risk bound.

6.6.3 *Collection of known results*

For completeness, we first give the definition of subgaussian norm, details could be found at Vershynin [2010].

Definition 59 (Subgaussian norm)**.** *The subgaussian norm $||X||_{\psi_2}$ of a subgaussian p-dimensional random vector X, is defined as*

$$||X||_{\psi_2} = \sup_{x\in\mathbb{S}^{p-1}}\sup_{q>1} q^{-1/2}(\mathbb{E}|\langle X, x\rangle|^q)^{1/q},$$

where $\mathbb{S}^{p-1}$ is the p-dimensional unit sphere.

We then define the restricted set $\mathcal{C}(|S|, 3)$ as

$$\mathcal{C}(|S|, 3) = \{\Delta \in \mathbb{R}^p | ||\Delta_{U^c}||_1 \leq 3||\Delta_U||_1, U \subset [p], |U| \leq |S|\}.$$

The following proposition is a simple extension of Theorem 6.2 in Bickel et al. [2009].

Proposition 60. *Let*

$$\lambda_t = A\sigma\sqrt{\frac{\log p}{n}}$$

with some constant $A > 2\sqrt{2}$ be the regularization parameter in lasso. With probability at least $1 - mp^{1-A^2/8}$,

$$||\widehat{\beta}_t - \beta_t^*||_1 \leq \frac{16A}{\kappa'} \sigma |S| \sqrt{\frac{\log p}{n}},$$

where κ is the minimum restricted eigenvalue of design matrix $X_1, \ldots, X_m$:

$$\kappa = \min_{t \in [m]} \min_{\Delta \in \mathcal{C}(|S|,3)} \frac{\Delta^T \left(\frac{X_t^T X_t}{n} \right) \Delta}{||\Delta_S||_2^2}.$$

Proof. Using Theorem 6.2 in Bickel et al. [2009] and take an union bound over $1, \ldots, m$ we obtain the result. $\square$

Lemma 61 (Equation (27) in Cavalier et al. [2002]; Lemma B.1 in Lounici et al. [2011])**.** *Let $\xi_1, \xi_2, \ldots \xi_m$ be i.i.d. standard normal random variables, let $v = (v_1, \ldots, v_m) \neq 0$, $\eta_v = \frac{1}{\sqrt{2}||v||_2} \sum_{i=1}^m (\xi_i^2 - 1) v_i$ and $m(v) = \frac{||v||_\infty}{||v||_2}$. We have, for all $t > 0$, that*

$$P(|\eta_v| > t) \leq 2 \exp\left(-\frac{t^2}{2 + 2\sqrt{2} t m(v)} \right).$$

The next lemma relies on the generalized coherence parameter:

Definition 62 (Generalized Coherence)**.** *For matrices $X \in \mathbb{R}^{n \times p}$ and $M = (m_1, \ldots, m_p) \in \mathbb{R}^{p \times p}$, let*

$$\mu(X, M) = \max_{j \in [p]} ||\Sigma m_j - e_j||_\infty$$

be the generalized coherence parameter between X and M, where $\Sigma = n^{-1} X^T X$. Furthermore, let $\mu^* = \min_{t \in [m]} \min_{M \in \mathbb{R}^{p \times p}} \mu(X_t, M)$ be the minimum generalized coherence.

Lemma 63 (Theorem 2.4 in Javanmard and Montanari [2014])**.** *When X_t are drawn from subgaussian random vectors with covariance matrix Σ_t, and $X_t \Sigma_t^{-1/2}$ has bounded subgaussian norm $||X_t \Sigma_t^{-1/2}||_{\psi_2} \leq \sigma_X$. When $n \geq 24 \log p$, then with probability at least $1 - 2p^{-2}$,* *we have*

$$\mu(X_t, \Sigma_t^{-1}) < 10e\sigma_X^4 \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} \sqrt{\frac{\log p}{n}}.$$

For subgaussian design, we also have the following restricted eigenvalue condition Rudelson and Zhou [2013], Lee et al. [2017b].

Lemma 64. *When X_t are drawn from subguassian random vectors with covariance matrix Σ_t, and bounded subgaussian norm σ_X. When $n \geq 4000 s' \sigma_X \log\left(\frac{60\sqrt{2}ep}{s'}\right)$ where $s' = \left(1 + 30000\frac{\lambda_{\max}}{\lambda_{\min}}\right)|S|$, and $p > s'$, then with probability at least $1 - 2\exp^{(-n/4000C_\kappa^4)}$, for any vector $\Delta \in \mathcal{C}(|S|, 3)$ where we have*

$$\Delta^T \left(\frac{X_t^T X_t}{n}\right) \Delta \geq \frac{1}{2}\lambda_{\min} ||\Delta_S||_2^2.$$

CHAPTER 7
DISTRIBUTED MULTI-TASK LEARNING WITH SHARED SUBSPACE

7.1 Motivation

Multi-task learning is widely used learning framework in which similar tasks are considered jointly for the purpose of improving performance compared to learning the tasks separately [Caruana, 1997]. By transferring information between related tasks it is hoped that samples will be better utilized, leading to improved generalization performance. Multi-task learning has been successfully applied, for example, in natural language understanding [Collobert et al., 2011], speech recognition [Seltzer and Droppo, 2013], remote sensing [Xue et al., 2007], image classification [Lapin et al., 2014], spam filtering [Weinberger et al., 2009], web search [Chapelle et al., 2010], disease prediction [Zhou et al., 2013], and eQTL mapping [Kim and Xing, 2010] among other applications.

Here, we study multi-task learning in a distributed setting, where each task is handled by a different machine and communication between machines is expensive. That is, each machine has access to data for a different task and needs to learn a predictor for that task, where machines communicate with each other in order to leverage the relationship between the tasks. This situation lies between a homogeneous distributed learning setting Shamir and Srebro [2014], where all machines have data from the same source distribution, and inhomogeneous consensus problems Ram et al. [2010], Boyd et al. [2011], Balcan et al. [2012] where the goal is to reach a single consensus predictor or iterate which is the same on all machines. The main argument for this setting is that if each machine indeed has access to different data (e.g. from a different geographical region or different types of users), as in the consensus problems studied by Balcan et al. [2012], then we should allow a different predictor for each distribution, instead of insisting on a single consensus predictor, while still trying to leverage the relationship and similarity between data distributions, as in classical multi-task learning. Heterogeneous distribution across machines is a nature phenomenon in some practical distributed learning problems. For example, the recent proposed federated learning [McMahan et al., 2017] framework tries to learn from decentralized data generated in user's model devices, where one of the main challenges is how to tackle the non i.i.d. distributions for different users. As was recently pointed out by Wang et al. [2016b], allowing separate predictors for each task instead of insisting on a consensus predictor changes the fundamental nature of the distributed learning problem, allows for different optimization methods, and necessitates a different analysis approach, more similar to homogeneous distributed learning as studied by Shamir and Srebro [2014].

The success of multi-task learning relies on the relatedness between tasks. While Wang et al. [2016b] studied tasks related through shared sparsity, here we turn to a more general, powerful and empirically more successful model of relatedness, where the predictors for different tasks lie in some (a-priori unknown) shared low-dimensional subspace and so the matrix of predictors is of low rank [Ando and Zhang, 2005, Amit et al., 2007, Yuan et al., 2007, Argyriou et al., 2008]. In a shared sparsity model, information from all tasks is used to learn a subset of the input features which are then used by all tasks. In contrast, in a shared subspace model, novel features, which are linear functions of the input features, are learned. The model can thus be viewed as a two-layer neural network, with the bottom layer learned jointly across tasks and the top layer task-specific. Being arguably the most complex multi-layer network that we can fully analyze, studying such models can also serve as a gateway to using deeper networks for learning shared representations.

Multi-task learning with a shared subspace is well-studied in a centralized setting, where data for all tasks are on the same machine, and some global centralized procedure is used to find a good predictor for each task. In such a situation, nuclear norm regularization is often used to leverage the low rank structure [e.g. Argyriou et al., 2008, Amit et al., 2007]

Approach	Rounds	Communication	Worker Comp.	Master Comp.
`Local`	1	0	ERM	0
`Centralize`	1	$\frac{A^2}{\epsilon_{\text{stat}}^2}\left(\frac{r}{m}+\frac{r}{\overline{p}}\right)$	0	Nuclear Norm Minimization
`SVD Truncation`	1	$2 \cdot p$	ERM	SV Shrinkage
`ProxGD`	$\frac{mHA^2}{\varepsilon}$	$2 \cdot p$	Gradient Comp.	SV Shrinkage
`AccProxGD`	$\sqrt{\frac{mHA^2}{\varepsilon}}$	$2 \cdot p$	Gradient Comp.	SV Shrinkage
`ADMM`	$\frac{mA^2}{\varepsilon}$	$3 \cdot p$	ERM	SV Shrinkage
`DFW`	$\frac{mHA^2}{\varepsilon}$	$2 \cdot p$	Gradient Comp.	Leading SV Comp.
`DGSP`	$\frac{mHA^2}{\varepsilon}$	$2 \cdot p$	ERM	Leading SV Comp.
`DNSP`	–	$2 \cdot p$	ERM	Leading SV Comp.

Table 7.1: Summary of resources required by different approaches to distributed multi-task learning with shared representations, in units of vector operations/communications, ignoring log-factors.

and learning guarantees are known ([Pontil and Maurer, 2013] and see also Section 7.2). With the growth of modern massive data sets, where tasks and data often too big to handle on a single machine, it is important to develop methods also for the distributed setting, and distribute learning of multiple related tasks have been an emerging technique studied recently [Vanhaesebrouck et al., 2016, Liu et al., 2017, Smith et al., 2017] . Unfortunately, we are not aware of any prior work on distributed multi-task learning with shared subspaces.

In this chapter we focus on methods with efficient communication complexity (i.e. with as small as possible communication between machines), that can still leverage most of the statistical benefit of shared-subspace multi-task learning. Although all our methods are also computationally tractable and can be implemented efficiently, we are less concerned here with minimizing the runtime on each machine separately, considering communication, instead, as the main bottleneck and the main resource to be minimized [Bekkerman et al., 2011]. This is similar to the focus in distributed optimization approaches such as ADMM [Boyd et al., 2011] and DANE [Shamir et al., 2014] where optimization within each machine is taken as an atomic step.

Contribution The main contributions of this chapter are:

i) We present and formalize the shared-subspace multi-task learning [Argyriou et al., 2008] in the novel distributed multitask setting, identifying the relevant problems and possible approaches. We analyze two baselines, several representative first-order distributed optimization methods, with careful sample and communication complexity analysis.

ii) We proposed and analyzed two subspace pursuit approaches which learn the shared representation in a greedy fashion, and leverage the low-dimensional predictive structure in a communication efficient way.

iii) We conducted comprehensive experimental comparisons of the discussed approaches on both simulated and real datasets, where we demonstrated that the proposed approaches are more communication efficient than first-order convex optimization methods.

Table 7.1 summarized the approaches studied in this chapter, which will be discussed in detail in the following sections.

7.2 Problem set-up and baselines

We consider a setting with m tasks, each characterized by a source distribution $\mathcal{D}_j(\mathbf{X}, Y)$ over feature vectors $\mathbf{X} \in \mathbb{R}^p$ and associated labels Y, and our goal is to find linear predictors $\mathbf{w}_1, \ldots, \mathbf{w}_m \in \mathbb{R}^p$ minimizing the overall expected loss (risk) across tasks:

$$\mathcal{L}(W) = \frac{1}{m} \sum_{j=1}^{m} \mathbb{E}_{(\mathbf{X}_j, Y_j) \sim \mathcal{D}_j} \left[\ell(\mathbf{w}_j^T \mathbf{X}_j, Y_j) \right], \tag{7.1}$$

where for convenience we denote $W \in \mathbb{R}^{p \times m}$ for the matrix with columns $\mathbf{w}_i$, and $\ell(\cdot, \cdot)$ is some specified instantaneous loss function.

In the learning setting, we cannot observe $\mathcal{L}(W)$ directly and only have access to i.i.d. sample $\{\mathbf{x}_{ji}, y_{ji}\}_{i=1}^{n_j}$ from each distribution $\mathcal{D}_j$, $j = 1, \ldots, m$. For simplicity of presentation, we will assume that $n_j = n$, $j = 1, \ldots, m$, throughout the chapter. We will denote the empirical loss $\mathcal{L}_n(W) = \frac{1}{m}\sum_{j=1}^{m} \mathcal{L}_{nj}(\mathbf{w}_j)$ where

$$\mathcal{L}_{nj}(\mathbf{w}_j) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}_j^T \mathbf{x}_{ji}, y_{ji})$$

is the local (per-task) empirical loss.

We consider a distributed setting, where each task is handled on one of m separate machines, and each machine j has access only to the samples drawn from $\mathcal{D}_j$. Communication between the machines is by sending real-valued vectors. Our methods work either in a broadcast communication setting, where at each iteration each machine sends a vector which is received by all other machines, or in a master-at-the-center topology where each machine sends a vector to the master node, whom in turn performs some computation and broadcasts some other vectors to all machines. Either way, we count the total number of vectors communicated.

7.2.1 Review of homogeneous, inhomogeneous and multi-task distributed learning.

We briefly review the relationship between homogeneous, inhomogeneous and multi-task learning, as recently presented by Wang et al. [2016b].

A typical situation considered in the literature is one in which data on different machines are all drawn i.i.d from the same source distribution. In this setting, tasks on different machines are all the same, which should be taken advantage of in optimization Shamir et al. [2014]. Furthermore, as each machine has access to samples from the source distribution it can perform computations locally, without ever communicating with other machines. While having zero communication cost, this approach does not compare favorably with the centralized approach, in which all data are communicated to the central machine and used to obtain one predictor, when measured in terms of statistical efficiency. The goal in this setting is to obtain performance close to that of the centralized approach, using the same number of samples, but with low communication and computation costs Shamir and Srebro [2014], Jaggi et al. [2014], Zhang et al. [2013c, 2012], Lee et al. [2017b]. Another setting considered in the distributed optimization literature is that of consensus optimization. Here each machine has data from a different distribution and the goal is to find one vector of coefficients that is good for all the separate learning or optimization problems Boyd et al. [2011], Ram et al. [2010], Balcan et al. [2012]. The difficulty of consensus problems is that the local objectives might be rather different, and, as a result, one can obtain lower bounds on the amount of communication that must be exchanged in order to reach a joint optimum.

In this chapter we suggest a novel setting that combines aspects of the above two settings. On one hand, we assume that each machine has a different source distributions $\mathcal{D}_j$, corresponding to a different task, as in consensus problems. For example, each machine serves a different geographical location, or each is at a different hospital or school with different characteristics. But if indeed there are differences between the source distributions, it is natural to learn different predictors $\mathbf{w}_j$ for each machine, so that $\mathbf{w}_j$ is good for the distribution typical to that machine. In this regard, our distributed multi-task learning problem is more similar to single-source problems, in that machines could potentially learn on their own given enough samples and enough time. Furthermore, availability of other machines just makes the problem easier by allowing transfer between the machine, thus reducing the sample complexity and potentially runtime. The goal, then, is to leverage as much transfer as possible, while limiting communication and runtime. As with single-source problems, we compare our method to the two baselines, where we would like to be much better than the local approach, achieving performance nearly as good as the centralized approach, but with minimal communication and efficient runtime.

As in standard agnostic-PAC type analysis, our goal will be to obtain expected loss $\mathcal{L}(W)$

which is not much larger than the expected loss of some (unknown) reference predictor[1] W^*, and we will measure the *excess error* over this goal. To allow obtaining such guarantees we will assume:

Assumption 65. *The loss function $\ell(\cdot,\cdot)$ is 1-Lipschitz and bounded[2] by 1, be twice differentiable and H-smooth, that is*

$$|\ell'(a,c) - \ell'(b,c)| \leq H|a-b|, \qquad \forall a,b,c \in \mathbb{R}.$$

All the data points are bounded by unit length, i.e. $||\mathbf{x}_{ji}||_2 \leq 1, \forall i,j$, and the reference predictors have bounded norm: $\max_{j\in[m]} ||\mathbf{w}_j^*||_2^2 \leq A^2$.

7.2.2 Baseline approaches

The simplest approach, which we refer to as `Local`, is to learn a linear predictor on each machine independently of other machines. This single task learning approach ignores the fact that the tasks are related and that sharing information between them could improve statistical performance. However, the communication cost for this procedure is zero, and with enough samples it can still drive the excess error to zero. However, compared to procedures discussed later, sample complexity (number of samples n required to achieve small excess error) is larger. A standard Rademacher complexity argument [Bartlett and Mendelson, 2002] gives the following generalization guarantee, which is an extension of Theorem 26.12 in Shalev-Shwartz and Ben-David [2014].

1. Despite the notation, W^* need *not* be the minimizer of the expected loss. We can think of it as the minimizer inside some restricted hypothesis class, though all analysis and statements hold for any chosen reference predictor W^*.

2. This is only required for the high probability bounds.

Proposition 66. *Suppose Assumption 65 holds. Then with probability at least* $1-\delta$*,*

$$\mathcal{L}(\widehat{W}_{\text{local}}) - \mathcal{L}(W^*) \leq \frac{2A}{\sqrt{n}} + \sqrt{\frac{2\ln(2m/\delta)}{n}},$$

where $\widehat{W}_{\text{local}} = [\widehat{\mathbf{w}}_1, \ldots, \widehat{\mathbf{w}}_m]$ *with* $\widehat{\mathbf{w}}_j = \arg\min_{||\mathbf{w}|| \leq A} \mathcal{L}_{nj}(\mathbf{w})$.

That is, in order to ensure ϵ_{stat} excess error, we need $n = \mathcal{O}\left(\frac{A^2}{\epsilon_{\text{stat}}^2}\right)$ samples from each task.

At the other extreme, if we ignore all communication costs, and, e.g. communicate all data to a single machine, we can significantly leverage the shared subspace. To understand this, we will first need to introduce two assumptions: one about the existence of a shared subspace (i.e. that the reference predictor is indeed low-rank), and the other about the spread of the data:

Assumption 67. $\text{rank}(W^*) \leq r$

Assumption 68. *There is a constant* $\widetilde{p}$*, such that*

$$\left|\left|\frac{1}{m}\sum_{j=1}^{m} \mathbb{E}_{(\mathbf{X}_j, Y_j)\sim\mathcal{D}_j}\left[\mathbf{X}_j\mathbf{X}_j^T\right]\right|\right|_2 \leq \frac{1}{\widetilde{p}}.$$

Since the data is bounded, we always have $1 \leq \widetilde{p} \leq p$, with $\widetilde{p}$ being a measure of how spread out the data is in different direction. A value of $1 = \widetilde{p}$ indicates the data is entirely contained in a one-dimensional line. In this case, the predictor matrix will also always be rank-one, imposing a low-rank structure is meaningless and we can't expect to gain from it. However, when $\widetilde{p}$ is close to p, or at least high, the data is spread in many directions and the low-rank assumption is meaningful. We can think of $\widetilde{p}$ as the "effective dimensionality" of the data, and hope to gain when $r \ll \widetilde{p}$.

With these two assumptions in hand, we can think of minimizing the empirical error subject to a rank constraint on W. This is a hard and non-convex optimization task, but we can instead use the nuclear norm (aka trace-norm) $||W||_*$ as a convex surrogate for the rank. This is because if Assumptions 65 and 67 hold, then we also have $||W^*||_* \leq \sqrt{rm}A$. With this in mind, we can define the following *centralized* predictor:

$$\widehat{W}_{\text{centralize}} = \arg \min_{||W||_* \leq \sqrt{rm}A} \mathcal{L}_n(W) \tag{7.2}$$

which achieves the improved excess error guarantee:

Proposition 69. *(Theorem 1 in Pontil and Maurer [2013]) Suppose Assumptions 65, 67 and 68 hold. Then with probability at least* $1 - \delta$*,*

$$\mathcal{L}(\widehat{W}_{\text{centralize}}) \leq \mathcal{L}(W^*) + \sqrt{\frac{2\ln(2/\delta)}{nm}} + 2\sqrt{r}A\left(\sqrt{\frac{1}{\widetilde{p}n}} + 5\sqrt{\frac{\ln(mn)+1}{mn}}\right)$$

The sample complexity per task, up to logarithmic factors, is thus only $n = \widetilde{\mathcal{O}}\left(\frac{A^2}{\epsilon^2_{\text{stat}}}\left(\frac{r}{m} + \frac{r}{\widetilde{p}}\right)\right)$. When $\widetilde{p} \gg m$, this is a reduction by a factor of r/m. That is, it is as if we needed to only learn r linear predictors instead of m.

The problem is that a naive computation of $\widehat{W}_{\text{centralize}}$ requires collecting all data on a single machine, i.e. communicating $O(n) = \widetilde{\mathcal{O}}\left(\frac{A^2}{\epsilon^2_{\text{stat}}}\left(\frac{r}{m} + \frac{r}{\widetilde{p}}\right)\right)$ samples per machine. In the next Sections, we aim at developing methods of approximating $\widehat{W}_{\text{centralized}}$ using communication efficient methods, or computing an alternate predictor with similar statistical properties but using much less communication.

One-shot SVD Truncation A natural question to ask is whether there exists a one-shot communication method for the shared representation problem considered here, that still matches the performance of centralized methods. One reasonable solution is to consider the SVD truncation approach, which is based on the following derivation: consider the well specified linear regression model:

$$\mathbf{y}_{ji} = \langle \mathbf{x}_{ji}, \mathbf{w}_j^* \rangle + \epsilon_{ji},$$

where ϵ_{ji} is drawn from mean-zero Gaussian noise. It is easy to verify the following equation for OLS estimation:

$$\widehat{\mathbf{w}}_{\text{local}(j)} = \mathbf{w}_j^* + \left(\sum_i \mathbf{x}_{ji} \mathbf{x}_{ji}^T \right)^{-1} \left(\sum_i \epsilon_{ji} \mathbf{x}_{ji} \right).$$

Since $\widehat{W}_{\text{local}}$ is just W^* plus some mean-zero Gaussian noise, it is natural to consider the following low-rank matrix denoising estimator:

$$\min_W ||\widehat{W}_{\text{local}} - W||_F^2 \quad \text{s.t.} \quad \text{rank}(W) = r.$$

where the solution is a simple SVD truncation, and can be implemented in a one-shot way: each worker send its `Local` solution to the master, which then performs an SVD truncation step to maintain the top-r components

$$\widehat{W}_{\text{svd}} = U_r S_r V_r^T, \quad \text{where} \quad USV^T = \texttt{SVD}(\widehat{W}_{\text{local}}),$$

and send the resulting estimation back to each worker, where U_r, S_r, V_r are top-r components of U, S, V. Though this approach might work well for some simple scenarios, but will generally fail when the features are highly correlated: although the `Local` solution $\widehat{W}_{\text{local}}$ can output normal estimation of W^*, the estimation noise $\left(\sum_i \mathbf{x}_{ji} \mathbf{x}_{ji}^T \right)^{-1} \left(\sum_i \epsilon_{ji} \mathbf{x}_{ji} \right)$ might be highly correlated (depend on the correlation between features), which makes the SVD truncation estimation not reliable.

7.3 Distributed convex optimization approaches

In this section, we study how to obtain the sharing benefit of the centralized approach using distributed convex optimization techniques, while keeping the communication requirements low.

To enjoy the benefit of nuclear-norm regularization while avoiding the high communication cost of `Centralize`, a flexible strategy is to solve the convex objective (7.2) via distributed optimization techniques. Let $W^{(t)}$ be the solution at t-iteration for some iterative distributed algorithms for (7.2), by the generalization error decomposition [Bousquet and Bottou, 2008],

$$\mathcal{L}(W^{(t)}) - \mathcal{L}(W^*) \leq 2\epsilon_{\text{stat}} + \epsilon_{\text{opt}},$$

Suppose $W^{(t)}$ satisfying $\mathcal{L}_n(W^{(t)}) \leq \mathcal{L}_n(\widehat{W}) + \mathcal{O}(\epsilon_{\text{opt}})$ with $\epsilon_{\text{opt}} = \mathcal{O}(\epsilon_{\text{stat}})$. Then $W^{(t)}$ will have the generalization error of order $\mathcal{O}(\epsilon_{\text{stat}})$. Therefore in order to study the generalization performance, we will study how the optimization error decreases as the function of the number of iterations t.

Distributed Proximal Gradient Maybe the simplest distributed optimization algorithm for (7.2) is the proximal gradient descent. It is not hard to see that computation of the gradient $\nabla \mathcal{L}_n(W)$ can be easily done in a distributed way as the losses are decomposable across machines:

$$\nabla \mathcal{L}_n(W) = \left[\nabla \mathcal{L}_{n1}(\mathbf{w}_1), \ldots, \nabla \mathcal{L}_{nm}(\mathbf{w}_m) \right]$$
$$\text{where} \quad \nabla \mathcal{L}_{nj}(\mathbf{w}_j) = \frac{1}{nm} \sum_{i=1}^{n} \ell'(\langle \mathbf{w}_j, \mathbf{x}_{ji} \rangle, y_{ji}) \mathbf{x}_{ji}.$$

Algorithm 17 `ProxGD`: Distributed Proximal Gradient.

```
for $t = 1, 2, \ldots$ do
    Workers:
    for $j = 1, 2, \ldots, m$ do
        Each worker compute the its gradient direction $\nabla \mathcal{L}_{nj}(\mathbf{w}_j^{(t)}) = \frac{1}{mn}\sum_{i=1}^{n} \ell'(\langle \mathbf{w}_j^{(t)}, \mathbf{x}_{ji}\rangle, y_{ji})\mathbf{x}_{ji}$, and send it to the master;
        Wait;
        Receive $\mathbf{w}_j^{(t+1)}$ from master.
    end
    Master:
    if Receive $\nabla \mathcal{L}_{nj}(\mathbf{w}_j^{(t)})$ from all workers then
        Concatenate the gradient vectors, and update $W^{(t+1)}$ as (7.3);
        Send $\mathbf{w}_j^{(t+1)}$ to all workers.
    end
end
```

Thus each machine j needs to compute the gradient $\nabla \mathcal{L}_{nj}(\mathbf{w}_j)$ on the local dataset and send it to the master. The master concatenates the gradient vectors to form the gradient matrix $\nabla \mathcal{L}_n(W)$. Finally, the master computes the proximal step

$$W^{(t+1)} = \arg\min_{W} ||W - (W^{(t)} - \eta \nabla \mathcal{L}_n(W^{(t)}))||_F^2 + \lambda ||W||_*, \tag{7.3}$$

which has the following closed form solution [Cai et al., 2010]: let $W^{(t)} - \eta \nabla \mathcal{L}_n(W^{(t)}) = U \Sigma V^T$ be the SVD of $W^{(t)} - \eta \nabla \mathcal{L}_n(W^{(t)})$, then $W^{(t+1)} = U\left(\Sigma - 0.5\lambda I\right)_+ V^T$ with $(x)_+ = \max\{0, x\}$ applied element-wise.

The algorithm is summarized in Algorithm 17 (in Appendix), which has well established convergence rates [Bach et al., 2012]:

$$\mathcal{L}_n(W^{(t)}) - \mathcal{L}_n(\widehat{W}) \leq \frac{mHA^2}{2t}.$$

Algorithm 18 `AccProxGD`: Accelerated Distributed Proximal Gradient for Multi-Task Learning.

```
for $t = 1, 2, \ldots$ do
    Workers:
    for $j = 1, 2, \ldots, m$ do
        Each worker compute the its gradient direction $\nabla \mathcal{L}_n(\mathbf{z}_j^{(t)}) = \frac{1}{mn} \sum_{i=1}^{n} \ell'(\langle \mathbf{z}_j^{(t)}, \mathbf{x}_{ji} \rangle, y_{ji}) \mathbf{x}_{ji}$, and send it to the master;
        Wait;
        Receive $\mathbf{z}_j^{(t+1)}$ from master.
    end
    Master:
    if Receive $\nabla \mathcal{L}_n(\mathbf{z}_j^{(t)})$ from all workers then
        Concatenate the gradient vectors, and update $W^{(t+1)}$ as (7.4);
        Update $Z^{(t+1)}$ as (7.5);
        Send $\mathbf{z}_j^{(t+1)}$ to all workers.
    end
end
```

To obtain ε-generalization error, the distributed proximal gradient descent requires $\mathcal{O}\left(\frac{mHA^2}{\varepsilon}\right)$ rounds of communication, with a total $\mathcal{O}\left(\frac{mHA^2 p}{\varepsilon}\right)$ bits communications per machine.

Distributed Accelerated Gradient It is also possible to use Nesterov's acceleration idea [Nesterov, 1983] to improve the convergence of the proximal gradient algorithm from $\mathcal{O}\left(\frac{1}{t}\right)$ to $\mathcal{O}\left(\frac{1}{t^2}\right)$ [Ji and Ye, 2009]. Using the distributed accelerated proximal gradient descent, one needs $\mathcal{O}\left(\sqrt{\frac{mHA^2}{\varepsilon}}\right)$ rounds of communication with a total $\mathcal{O}\left(\sqrt{\frac{mHA^2}{\varepsilon}} \cdot p\right)$ bits communicated per machine to achieve ε-generalization error. The algorithm is summarized in Algorithm 18 (in Appendix), where the master maintains two sequences: W and Z. First, a proximal gradient update of W is done based on Z

$$W^{(t+1)} = \arg\min_{Z} ||Z - (Z^{(t)} - \eta \nabla \mathcal{L}_n(Z^{(t)}))||_F^2 + \lambda ||Z||_* \tag{7.4}$$

and then Z is updated based on a combination of the current W and the difference with previous W

$$Z^{(t+1)} = W^{(t+1)} + \gamma_t(W^{(t+1)} - W^{(t)}). \tag{7.5}$$

7.3.1 Distributed alternating direction methods of multipliers

Algorithm 19 `ADMM`: Distributed ADMM for Multi-Task Learning.

```
for $t = 1, 2, \dots$ do
    Workers:
    for $j = 1, 2, \dots, m$ do
        Each worker solves the regularized ERM problem as (7.6) to get $\mathbf{w}_j^{(t+1)}$, and send it to the master;
        Wait;
        Receive $\mathbf{z}_j^{(t+1)}, \mathbf{q}_j^{(t+1)}$ from master.
    end
    Master:
    if Receive $\mathbf{w}_j^{(t+1)}$ from all workers then
        Concatenate the current solutions $\mathbf{w}_j^{(t+1)}$, and update $Z^{(t+1)}$ as (7.7);
        Update $Q^{(t+1)}$ as (7.8);
        Send $\mathbf{z}_j^{(t+1)}, \mathbf{q}_j^{(t+1)}$ to the corresponding worker.
    end
end
```

The Alternating Direction Methods of Multipliers (ADMM) is also a popular method for distributed optimization [Boyd et al., 2011] and can be used to solve the distributed low-rank multi-task learning problem. We first write the objective (7.2) as

$$\arg\min_{W,Z} \; \mathcal{L}_n(W) + \lambda||Z||_*, \quad \text{subject to} \quad W = Z.$$

By introducing the Lagrangian and augmented terms, we get the following unconstrained problem:

$$\begin{aligned}\widetilde{\mathcal{L}}(W, Z, Q) =& \mathcal{L}_n(W) + \lambda||Z||_* + \langle W - Z, Q\rangle \\ &+ \frac{\rho}{2}||W - Z||_F^2,\end{aligned}$$

where ρ is a parameter controlling the augmentation level. Note that except for Z, the augmented Lagrangian objective are decomposable across tasks. To implement the distributed ADMM algorithm, we let the workers maintain the data and W, while the master maintains Z and Q. At round t, each machine separately solves

$$\begin{aligned}\mathbf{w}_j^{(t+1)} =& \arg\min_{\mathbf{w}} \mathcal{L}_{nj}(\mathbf{w}_j) + \langle \mathbf{w}_j^{(t+1)} - \mathbf{z}_j^{(t)}, \mathbf{q}_j^{(t)}\rangle \\ &+ \frac{\rho}{2}||\mathbf{w}_j^{(t+1)} - \mathbf{z}_j^{(t)}||_2, \end{aligned} \tag{7.6}$$

which is minimizing the local loss plus a regularization term. Next, each worker sends their solution to the master, which performs the following updates for Z and Q

$$\begin{aligned}Z^{(t+1)} =& \arg\min_{Z} \ \langle W^{(t+1)} - Z, Q^t\rangle + \lambda||Z||_* \\ &+ \frac{\rho}{2}||W^{(t+1)} - Z||_F^2, \end{aligned} \tag{7.7}$$

$$Q^{(t+1)} = Q^{(t)} + \rho(W^{(t+1)} - Z^{(t+1)}), \tag{7.8}$$

which have closed-form solutions.

The algorithm `ADMM` is summarized in Algorithm 19. Note that compared to methods discussed before, `ADMM` needs to communicate three p-dimensional vectors between each worker and the master at each round, while the proximal gradient approaches only communicate two p-dimensional vectors per round. Based on convergence results of ADMM [He and Yuan, 2012], $\mathcal{O}\left(\frac{mA^2}{\varepsilon}\right)$ rounds of communication are needed to obtain ε-generalization error.

Algorithm 20 DFW: Distributed Frank-Wolfe for Multi-Task Learning.

```
for $t = 0, 2, \dots$ do
    Workers:
    for $j = 1, 2, \dots, m$ do
        Each worker compute the its gradient direction $\nabla \mathcal{L}_{nj}(\mathbf{w}_j^{(t)})$, and send it to the master;
    end
    if Receive $\mathbf{v}_j \mathbf{u}$ from the master then
        Set $\gamma = \frac{2}{t+2}$;
        Update $\mathbf{w}_j^{(t+1)}$ as (7.9).
    end
    Master:
    if Receive $\nabla \mathcal{L}_{nj}(\mathbf{w}_j^{(t)})$ from all workers then
        Concatenate the gradient vectors, and compute the largest singular vectors: $(\mathbf{u}, \mathbf{v}) = \mathsf{SV}(\nabla \mathcal{L}_n(W^{(t)}))$;
        Send $\mathbf{v}_j \mathbf{u}$ to $j$-th worker.
    end
end
```

7.3.2 Distributed Frank-Wolfe method

Another approach we consider is the distributed Frank-Wolfe method [Frank and Wolfe, 1956, Jaggi, 2013, Bellet et al., 2015]. This methods does not require performing SVD, which might bring additional computational advantages. Instead of directly minimizing the nuclear norm regularized objective, the Frank-Wolfe algorithm considers the equivalent constrained minimization problem

$$\min_W \mathcal{L}_n(W) \quad \text{subject to} \quad ||W||_* \leq R.$$

At each step, Frank-Wolfe algorithm considers the following direction to update

$$Z^{(t)} = \arg \min_{||Z||_* \leq R} \langle \nabla \mathcal{L}_n(W^{(t)}), Z \rangle = -R \cdot \mathbf{u}\mathbf{v}^T,$$

where $(\mathbf{u}, \mathbf{v}) = \mathsf{SV}(\nabla \mathcal{L}_n(W^{(t)}))$ is the leading singular vectors of $\nabla \mathcal{L}_n(W^{(t)})$. The next iterate is obtained as

$$W^{(t+1)} = (1-\gamma)W^{(t)} + \gamma Z^{(t)},$$

where γ is a step size parameter. To implement this algorithm in a distributed way, the master first collects the gradient matrix $\nabla \mathcal{L}_n(W^{(t)})$ and computes $\mathbf{u}$ and $\mathbf{v}$. The vector $\mathbf{v}_j \mathbf{u}$ is sent to j-th machine, which performs the following update:

$$\mathbf{w}_j^{(t+1)} = (1-\gamma)\mathbf{w}_j^{(t)} - \gamma R \mathbf{v}_j \mathbf{u}. \tag{7.9}$$

The algorithm is summarized in Algorithm 20. Similar to the distributed (accelerated) proximal gradient descent, the distributed Frank-Wolfe only requires communication of two p-dimensional vectors per round. Though computationally cheaper compared to other methods considered in this section, the distributed Frank-Wolfe algorithm enjoys similar convergence guarantees to the distributed proximal gradient descent [Jaggi, 2013], that is, after $\mathcal{O}\left(\frac{mHA^2}{\varepsilon}\right)$ iterations, the solution will be ε suboptimal.

7.4 Distributed greedy representation learning approaches

In this section we propose two distributed algorithms which select the subspaces in a greedy fashion, instead of solving the nuclear norm regularized convex program.

7.4.1 Distributed gradient subspace pursuit

Our greedy approach is inspired by the methods used for sparse signal reconstruction [Tropp, 2004, Shalev-Shwartz et al., 2010]. Under the assumption that the optimal model W^* is low-

Algorithm 21 DGSP: Distributed Gradient Subspace Pursuit.

```
for $t = 1, 2, \ldots$ do
    Workers:
    for $j = 1, 2, \ldots, m$ do
        Each worker compute the its gradient direction $\nabla \mathcal{L}_{nj}(\mathbf{w}_j^{(t)})$, and send it to the master.
    end
    if Receive $\mathbf{u}$ from the master then
        Update the projection matrix $U = [U \ \mathbf{u}]$;
        Solve the projected ERM problem: $\mathbf{v}_j = \arg\min_{\mathbf{v}_j} \mathcal{L}_{nj}(U\mathbf{v}_j)$;
        Update $\mathbf{w}_j^{(t+1)} = U\mathbf{v}_j$.
    end
    Master:
    if Receive $\nabla \mathcal{L}_{nj}(\mathbf{w}_j^{(t)})$ from all workers then
        Concatenate the gradient vectors, and compute the largest singular vectors: $(\mathbf{u}, \mathbf{v}) = \mathsf{SV}(\nabla \mathcal{L}_n(W^{(t)}))$;
        Send $\mathbf{u}$ to all workers.
    end
end
```

rank, say rank r, we can write W^* as a sum of r rank-1 matrices:

$$W^* = \sum_{i=1}^{r} a_i \mathbf{u}_i \mathbf{v}_i^T = UV^T,$$

where $a_i \in \mathbb{R}, \mathbf{u}_i \in \mathbb{R}^p, \mathbf{v}_i \in \mathbb{R}^m$, and $||\mathbf{u}_i||_2 = ||\mathbf{v}_i||_2 = 1$. In the proposed approach, the projection matrix U is learned in a greedy fashion. At every iteration, a new one-dimensional subspace is identified that leads to an improvement in the objective. This subspace is then included into the existing projection matrix. Using the new expanded projection matrix as the current feature representation, we refit the model to obtain the coefficient vectors V. In the distributed setting, there is a master that gathers local gradient information from each task. Based on this information, it then computes the subspace to be added to the projection matrix and sends it to each machine. The key step in the distributed greedy subspace pursuit algorithm is the addition of the subspace. One possible choice is the principle component of the gradient direction; after the master collected the gradient matrix $\nabla\mathcal{L}_n(W^{(t)})$, it computes the top left and right singular vectors of $\nabla\mathcal{L}_n(W^{(t)})$. Let $(\mathbf{u},\mathbf{v}) = \mathsf{SV}(\nabla\mathcal{L}_n(W^{(t)}))$ be the largest singular vectors of $\nabla\mathcal{L}_n(W^{(t)})$. The left singular vector $\mathbf{u}$ is used as a new subspace to be added to the projection matrix U. This vector is sent to each machine, which then concatenate it to the projection matrix and refit the model with the new representation. Algorithm 21 details the steps.

Distributed gradient subspace pursuit (`DGSP`), detailed in Algorithm 21, creates subspaces that are orthogonal to each other, as shown in the following proposition which is proved in Appendix 7.6.1:

Proposition 70. *At every iteration of Algorithm 21, the columns of U are orthonormal.*

Both the distributed gradient subspace pursuit and the distributed Frank-Wolfe use the leading singular vector of the gradient matrix iteratively. Moreover, leading singular vectors of the gradient matrix have been used in greedy selection procedures for solving low-rank matrix learning problems [Shalev-Shwartz et al., 2011a, Wang et al., 2015]. However, `DGSP` utilize the learned subspace in a very different way: `GECO` [Shalev-Shwartz et al., 2011a] re-fit the low-rank matrix under a larger subspace which is spanned by all left and right singular vectors; while `OR1MP` [Wang et al., 2015] only adjust the linear combination parameters $\{a_i\}_{i=1}^r$ of the rank-1 matrices. The `DGSP` algorithm do not restrict on the joint subspaces $\{\mathbf{u}_i\mathbf{v}_i^T\}$, but focused on the low-dimensional subspace induced the projection matrix U, and estimate the task specific predictors V based on the learned representation.

Next, we present convergence guarantees for the distributed gradient subspace pursuit. First, note that the smoothness of $\ell(\cdot,\cdot)$ implies the smoothness property for any rank-1 update.

Proposition 71. *Suppose Assumption 65 holds. Then for any W and unit length vectors $\mathbf{u}\in\mathbb{R}^p$ and $\mathbf{v}\in\mathbb{R}^m$, we have $\mathcal{L}_n(W+\eta\mathbf{u}\mathbf{v}^T) \le \mathcal{L}_n(W) + \mathbf{u}^T\nabla\mathcal{L}_n(W)\mathbf{v} + \frac{H\eta^2}{2}$.*

We defer the proof in Appendix 7.6.2. The following theorem states the number of iterations needed for the distributed gradient subspace pursuit to find an ε-suboptimal solution.

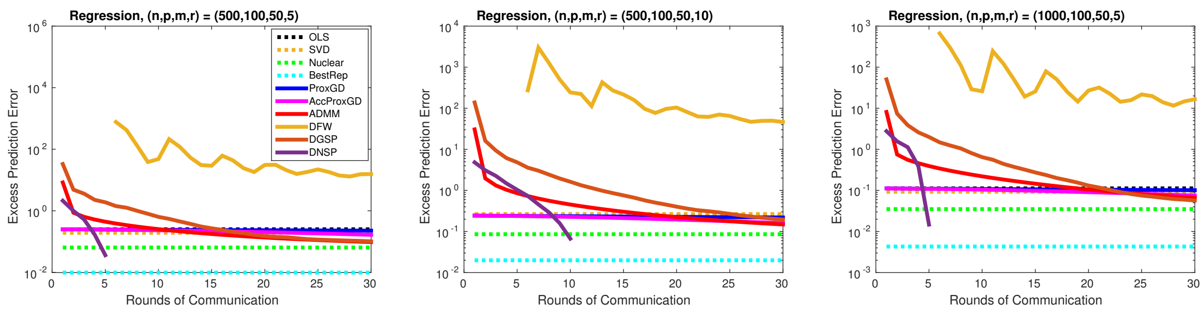


Figure 7.1: Excess prediction error for multi-task regression, with highly correlated features.

Theorem 72. *Suppose Assumption 65 holds. Then the distributed gradient subspace pursuit finds $W^{(t)}$ such that $\mathcal{L}_n(W^{(t)}) \leq \mathcal{L}_n(\widehat{W}) + \varepsilon$ when $t \geq \left\lceil \frac{4HmA^2}{\varepsilon} \right\rceil$.*

We defer the proof in Appendix 7.6.3. Theorem 72 tells us that for the distributed gradient subspace pursuit requires $\mathcal{O}\left(\frac{mHA^2}{\varepsilon}\right)$ iterations to reach ϵ accuracy. Since each iteration requires communicating p number, the communication cost per machine is $\mathcal{O}\left(\frac{mHA^2}{\varepsilon} \cdot p\right)$.

Improved convergence under restricted strong convexity The rates established in Theorem 72 can be further improved to have linear convergence is the restrict strong convexity holds, as recently showed in [Khanna et al., 2017]. Under such a stronger condition the distributed proximal gradient type methods will also enjoy linear convergence [Agarwal et al., 2012].

7.4.2 Distributed Newton subspace pursuit

In some applications the communication cost of `DNSP` might be still too high and in order to improve it we will try to reduce the number of rounds of communication. To that end, we develop a procedure that utilizes the second-order information to improve the convergence. Algorithm 22 describes the Distributed Newton Subspace Pursuit algorithm (`DNSP`). Note that distributed optimization with second-order information have been studied recently to achieve communication efficiency [Shamir et al., 2014, Zhang and Xiao, 2015].

Algorithm 22 DNSP: Distributed Newton Subspace Pursuit.

```
for $t = 1, 2, \dots$ do
    Workers:
    for $j = 1, 2, \dots, m$ do
        Each worker computes the Newton direction $\Delta\mathcal{L}_{nj}(\mathbf{w}_t^{(t)}) = \left(\nabla^2\mathcal{L}_{nj}(\mathbf{w}_t^{(t)})\right)^{-1} \nabla\mathcal{L}_{nj}(\mathbf{w}_t^{(t)})$ and sends it to the master.
    end
    if Receive $\mathbf{u}$ from the master then
        Perform Gram-Schmidt orthogonalization:
        $\mathbf{u} \leftarrow \mathbf{u} - \sum_{k=1}^{t-1}\langle U_k, \mathbf{u}\rangle$;
        Normalize $\mathbf{u} = \mathbf{u}/||\mathbf{u}||_2$;
        Update the projection matrix $U = [U \ \mathbf{u}]$;
        Solve the projected ERM problem:
        $\mathbf{v}_j = \arg\min_{\mathbf{v}_j} \frac{1}{n}\sum_{i=1}^{n} \ell(\langle \mathbf{v}_j, U^T X_{ji}\rangle, y_{ji})$;
        Update $\mathbf{w}_j^{(t+1)} = U\mathbf{v}_j$.
    end
    Master:
    if Receive $\Delta\mathcal{L}_{nj}(\mathbf{w}_t^{(t)})$ from all workers then
        Concatenate the Newton vectors, and compute the largest singular vectors: $(\mathbf{u}, \mathbf{v}) = \mathsf{SV}(\Delta\mathcal{L}_n(W^{(t)}))$;
        Send $\mathbf{u}$ to all workers.
    end
end
```

Compared to the gradient based methods, the DNSP algorithm uses second-order information to find subspaces to work with. At each iteration, each machine computes the Newton direction

$$\begin{aligned}\Delta\mathcal{L}_{nj}(\mathbf{w}_j) =& [\nabla^2\mathcal{L}_{nj}(\mathbf{w}_j)]^{-1}\nabla\mathcal{L}_{nj}(\mathbf{w}_j) \\ =& \left[\frac{1}{mn}\sum_{i=1}^{n} \ell''(\mathbf{w}_j^T\mathbf{x}_{ji}, y_{ji})\mathbf{x}_{ji}\mathbf{x}_{ji}^T\right]^{-1} \nabla\mathcal{L}_{nj}(\mathbf{w}_j),\end{aligned}$$

based on the current solution and sends it to the master. The master computes the overall Newton direction by concatenating the Newton direction for each task

$$\Delta\mathcal{L}_n(W) = [\Delta\mathcal{L}_{n1}(\mathbf{w}_1), \Delta\mathcal{L}_{n2}(\mathbf{w}_2), \dots, \Delta\mathcal{L}_{nm}(\mathbf{w}_m)]$$

and computes the top singular vectors of $\Delta\mathcal{L}_n(W)$. The top left singular vector $\mathbf{u}$ is then sent back to every machine, which is then concatenated to the current projection matrix. Each machine re-fits the predictors using the new representation. Note that at every iteration a Gram-Schmidt step is performed to ensure that the learned basis are orthonormal.

DNSP is a Newton-like method which uses second-order information, thus its generic analysis is not immediately apparent. However empirical results in the next section illustrate good performance of the proposed DNSP.

7.5 Experiments

We first illustrate performance of different procedures on simulated data. We generate data according to

$$
\begin{aligned}
&y_{ji} \mid \mathbf{x}_{ji} \sim \mathcal{N}(\mathbf{w}_j^T \mathbf{x}_{ji}, 1), \quad \text{and} \\
&y_{ji} \mid \mathbf{x}_{ji} \sim \text{Bernoulli}\left(\left(1 + \exp(-\mathbf{w}_j^T \mathbf{x}_{ji})\right)^{-1}\right)
\end{aligned}
$$

for regression problems and for classification problems, respectively. We generate the low-rank W^* as follows. We first generate two matrices $A \in \mathbb{R}^{p\times r}, B \in \mathbb{R}^{m\times r}$ with entries sampled independently from a standard normal distribution. Then we extract the left and right singular vectors of AB^T, denoted as U, V. Finally, we set $W^* = USV^T$, where S is a diagonal matrix with exponentially decaying entries: $\text{diag}(S) = [1, 1/1.5, 1/(1.5)^2, \ldots, 1/(1.5)^r]$. The feature vectors $\mathbf{x}_{ji}$ are sampled from a mean zero multivariate normal with the covariance matrix $\Sigma = (\Sigma_{ab})_{a,b\in[p]}$, $\Sigma_{ab} = 2^{-0.1|a-b|}$. The regularization parameters for all approaches were optimized to give the best prediction performance over a held-out validation dataset. For ProxGD and AccProxGD, we initialized the solution from Local. Our simulation results are averaged over 10 independent runs.

We investigate how the performance of various procedures changes as a function of problem parameters (n, p, m, r). We compare the approaches listed in Table 7.1. The results are shown in Figure 7.1 [3], respectively. We plot how the excess prediction error decreases as the number of rounds of communications increases (`Local`, `Centralize` and `BestRep` are one shot approaches thus the lines are horizontal). From the plots, we have the following observations:

i) One-shot SVD truncation approach does not significantly outperforms `Local`, sometimes even slightly worse.

ii) Nuclear norm regularization boosts the prediction performance over plain single task learning significantly, which shows clear advantage of leveraging the shared representation in multi-task learning.

iii) `ADMM` and `AccProxGD` perform reasonably well, especially `ADMM`. One reason for the effectiveness of `ADMM` is that for the problem of nuclear norm regularized multi-task learning considered here, the `ADMM` update solves regularized ERM problems at every iteration. `ADMM` and `AccProxGD` clearly outperform `ProxGD`.

iv) `ProxGD` and `DGSP` perform similarly. `DGSP` usually becomes worse as the iterations increases, while `ProxGD` converges to a global optimum of the nuclear norm regularized objective.

v) `DNSP` is the most communication-efficient method, and usually converges to a solution that is slightly better compared to the optimum of the nuclear regularization. This shows that second-order information helps a lot in reducing the communication cost.

vi) The `DFW` performs the worst in most cases, even though `DFW` shares some similarity with `DGSP` in learning the subspace. The empirical results suggest the re-fitting step in `DGSP` is very important.

3. For better visualization, here we omit the plot for `DFW` as its performance is significantly worse than others.

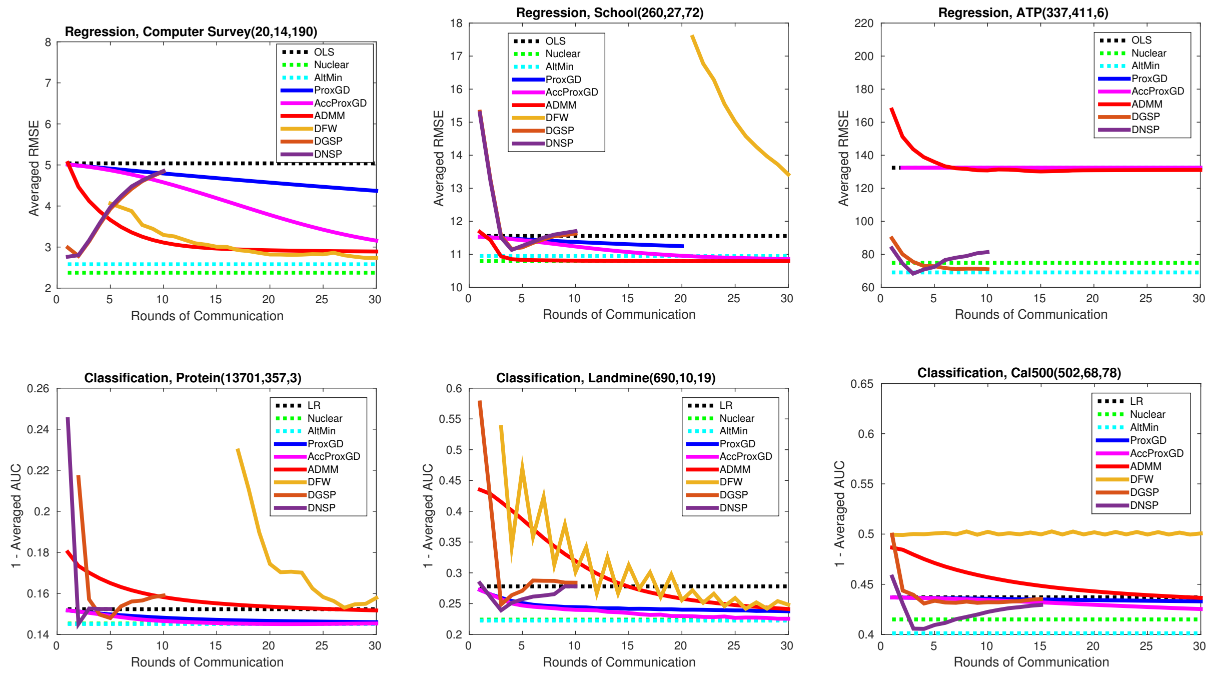


Figure 7.2: Prediction Error on real data.

7.5.1 Evaluation on real world datasets

We also evaluate discussed algorithms on several real world data sets, with 20% of the whole dataset as training set, 20% as held-out validation, then report the testing performance on the remaining 60%. For the real data, we have observed that adding ℓ_2 regularization usually helps improving the generalization performance. For the `Local` procedure we added an ℓ_2 regularization term (leads to ridge regression or ℓ_2 regularized logistic regression). For `DGSP` and `DNSP`, we also add an ℓ_2 regularization in finding the subspaces and refitting. We have worked on the following multi-task learning datasets:

School.[4] The dataset consists of examination scores of students from London's secondary schools during the years 1985, 1986, 1987. There are 27 school-specific and student-specific features to describe each student. The instances are divided by different schools, and the task is to predict the students' performance. We only considered schools with at least 100 records,

4. `http://cvn.ecp.fr/personnel/andreas/code/mtl/index.html`

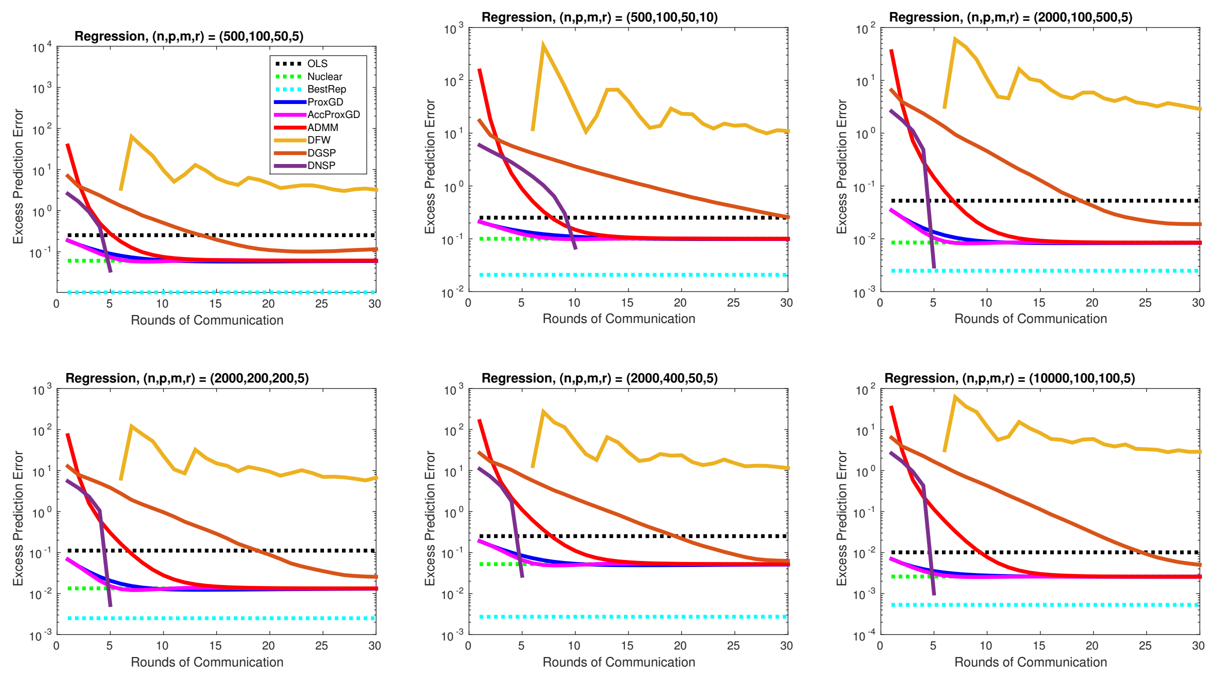


Figure 7.3: Excess prediction error for multi-task regression.

which results in 72 tasks in total. The maximum number of records for each individual school is 260.

Computer Survey. The data is taken from a conjoint analysis experiment [Lenk et al., 1996] which surveyed 180 persons about the probability of purchasing 20 kinds of personal computers. There are 14 variables for each computer, the response is an integer rating with scale $0 - 10$.

ATP.[5] The task here is to predict the airline ticket price [Spyromitros-Xioufis et al., 2016]. We are interested in the minimum prices next day for some specific observation date and departure date pairs. Each case is described by 411 features, and there are 6 target minimum prices for different airlines to predict. The sample size is 337.

Protein. Given the amino acid sequence, we are interested predicting the protein secondary structure [Sander and Schneider, 1991]. We tackle the problem by considering the

5. `http://mulan.sourceforge.net/datasets.html`

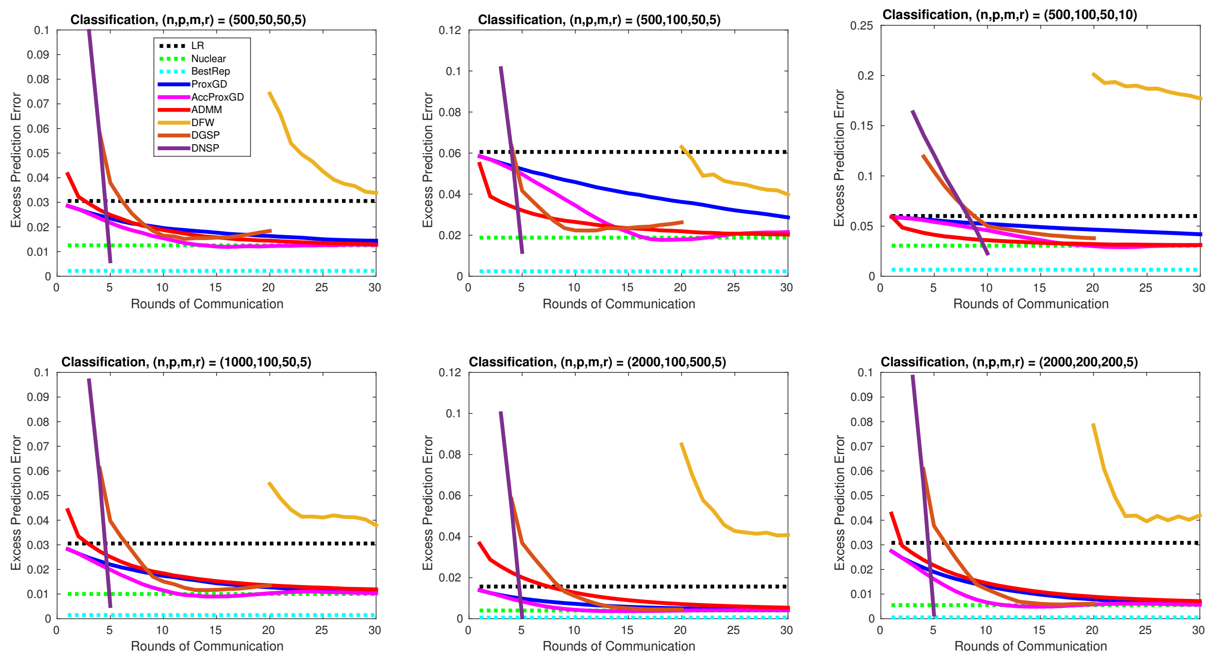


Figure 7.4: Excess prediction error for multi-task classification.

following three binary classification tasks: coil vs helix, helix vs strand, strand vs coil. Each sequence is described by 357 features. There are 24,387 instances in total.

Landmine. The data is collected from 19 landmine detection tasks [Xue et al., 2007]. Each landmine field is represented by a 9-dimensional vector extracted from radar images, containing moment-based, correlation-based, energy ratio, and spatial variance features. The sample size for each task varies from 445 to 690.

Cal500.[6] This music dataset [Turnbull et al., 2008] consists of 502 songs, where for each song 68 features are extracted. Each task is to predict whether a particular musically relevant semantic keyword should be an annotation for the song. We only consider tags with at least 50 times apperance, which results in 78 prediction tasks.

We compared various approaches as in the simulation study, except the `BestRep` as the best low-dimensional representation is unknown. We also compared with `AltMin`, which learns low-rank prediction matrix using the alternating minimization [Jain et al., 2013]. The

6. `http://eceweb.ucsd.edu/~gert/calab/`

results are shown in Figure 7.2. Since the labels for the real world classification datasets are often unbalanced, we report averaged area under the curve (AUC) instead of classification accuracy. We have the following observations:

- The distributed first-order approaches converge much slower than in simulations, especially on ATP and Cal500. We suspect this is because in the simulation study, the generated data are usually well conditioned, which makes faster convergence possible for such methods [Agarwal et al., 2012, Hong and Luo, 2017]. On real data, the condition number can be much worse.

- In most case, `DNSP` is the best in terms of communication-efficiency. `DGSP` also has reasonable performance with fewer round of communications compared to distributed first-order approaches.

- Among the first-order distributed convex optimization methods, `AccProxGD` is overall the most communication-efficient, while `DFW` is the worst, though it might have some advantages in terms of computation. Also, we observed significant zig-zag behavior of the `DFW` algorithm, as discussed in [Lacoste-Julien and Jaggi, 2015].

7.6 Proofs of technical results

7.6.1 Proof of Proposition 70

Proof. It is sufficient to prove that at every iteration, the current projection matrix U and the subspace to be added $\mathbf{u}$ are orthogonal to each other. Note that by the optimality condition:

$$\nabla_V \left(\mathcal{L}_n(UV^T) \right) = U^T \nabla \mathcal{L}_n(W^{(t)}) = 0.$$

Since $\mathbf{u}$ is the leading left singular vector of $\nabla\mathcal{L}_n(W^{(t)})$, we have $U^T\mathbf{u} = 0$. Each column of U has unit length, since it is a left singular vector of some matrix. $\square$

7.6.2 Proof of Proposition 71

Proof. It is sufficient to prove that the largest eigenvalue of $\nabla^2\mathcal{L}_n(W)$ does not exceed H. Since $\nabla^2\mathcal{L}_n(W)$ is a block diagonal matrix, it is sufficient to show that for every block $j \in [m]$, the largest eigenvalue of the block $\nabla^2\mathcal{L}_{nj}(\mathbf{w}_j)$ is not larger than H.

This is true by the H-smoothness of $\ell(\cdot)$ and the fact that the data points have bounded length:

$$||\nabla^2\mathcal{L}_{nj}(\mathbf{w}_j)||_2 \le H \cdot \max_{i,j} ||\mathbf{x}_{ji}||_2 \le H.$$

$\square$

7.6.3 Proof of Theorem 72

Proof. Let $(\mathbf{u}, \mathbf{v}) = \mathsf{SV}(\nabla\mathcal{L}_n(W^{(t)}))$, without loss of generality we can assume $\mathbf{u}^\top\nabla\mathcal{L}_n(W^{(t)})\mathbf{v} < 0$ (otherwise let $\mathbf{u} = -\mathbf{u}$), by the smoothness of $\mathcal{L}_n$, we know

$$\begin{aligned}
\mathcal{L}_n(W^{(t+1)}) \le & \min_b \mathcal{L}_n(W^{(t)} + b\mathbf{u}\mathbf{v}^T) \\
\le & \mathcal{L}_n(W^{(t)}) + b\langle \mathbf{u}\mathbf{v}^T, \nabla\mathcal{L}_n(W^{(t)})\rangle + \frac{Hb^2}{2} \\
\le & \mathcal{L}_n(W^{(t)}) + \frac{b\langle \widehat{W}, \nabla\mathcal{L}_n(W^{(t)})\rangle}{||\widehat{W}||_F} + \frac{Hb^2}{2}.
\end{aligned}$$

Let $W^{(t)} = UV^T$. Since V is a minimizer of $\mathcal{L}_n(UV^T)$ with respect to V, we have $U^T\nabla\mathcal{L}_n(W^{(t)}) = 0$ and therefore $\langle W^{(t)}, \nabla\mathcal{L}_n(W^{(t)})\rangle = \mathrm{trace}(VU^T\nabla\mathcal{L}_n(W^{(t)})) = 0$. From convexity of $\mathcal{L}_n(\cdot)$, we have

$$\begin{aligned}\langle \widehat{W}, \nabla \mathcal{L}_n(W^{(t)})\rangle =& \langle \widehat{W} - W^{(t)}, \nabla \mathcal{L}_n(W^{(t)})\rangle \\ \leq& \mathcal{L}_n(\widehat{W}) - \mathcal{L}_n(W^{(t)}).\end{aligned}$$

Combining with the display above

$$\mathcal{L}_n(W^{(t)}) - \mathcal{L}_n(W^{(t+1)}) \geq \frac{b(\mathcal{L}_n(W^{(t)}) - \mathcal{L}_n(\widehat{W}))}{||\widehat{W}||_F} - \frac{Hb^2}{2}.$$

By choosing

$$b = \frac{\mathcal{L}_n(W^{(t)}) - \mathcal{L}_n(\widehat{W})}{H||\widehat{W}||_F}$$

we have

$$\begin{aligned}\mathcal{L}_n(W^{(t)}) - \mathcal{L}_n(W^{(t+1)}) \geq& \frac{\left(\mathcal{L}_n(W^{(t)}) - \mathcal{L}_n(\widehat{W})\right)^2}{2H||\widehat{W}||_F^2} \\ \geq& \frac{\left(\mathcal{L}_n(W^{(t)}) - \mathcal{L}_n(\widehat{W})\right)^2}{2mHA^2}.\end{aligned}$$

Using Lemma 73 in Appendix, and let $\varepsilon_t = \mathcal{L}_n(W^{(t)}) - \mathcal{L}_n(\widehat{W})$, we know that after

$$t \geq \left\lceil \frac{2mHA^2}{\varepsilon} \right\rceil$$

iterations, we have $\mathcal{L}_n(W^{(t)}) \leq \mathcal{L}_n(\widehat{W}) + \varepsilon$. □

7.6.4 An auxiliary lemma

Lemma 73. *(Lemma B.2 of Shalev-Shwartz et al. [2010]) Let $x > 0$ and let $\varepsilon_0, \varepsilon_1, ...$ be a sequence such that $\varepsilon_{t+1} \leq \varepsilon_t - c\varepsilon_t^2$ for all t. Let ε be a positive scalar and t be a positive integer such that $t \geq \lceil \frac{1}{c\varepsilon} \rceil$. Then $\varepsilon_t \leq \varepsilon$.*

CHAPTER 8
CONCLUSION AND FUTURE DIRECTIONS

In this chapter, we summarize the key contributions made in this thesis to the field of distributed machine learning, and then discuss several open questions for each work. Finally, we outline several interesting research directions.

In Chapter 2, we proposed a novel approach for distributed learning with sparsity, which is efficient in both computation and communication. Our theoretical analysis showed that the proposed method works under weaker conditions than AvgDebias estimator while matches its error bound with one round communication. Furthermore, the estimation error can be improved with a logarithmic more rounds of communication until matching the centralized procedure. As we see in real data experiments, the proposed approach can still perform slightly worse than the centralized approach on certain datasets.

In Chapter 3, we made progress toward linear speedup, communication and memory efficient methods for distributed stochastic optimization, although we still do not have an algorithm that obtains the "ideal" distributed stochastic optimization performance of linear speedup with constant or near constant communication and memory. There is also no single known algorithm that dominates all others, with different methods being preferable in terms of different resources.

In Chapter 4, we focused on sketching techniques for solving large-scale ℓ_2 regularized least square problems. We established the equivalence between two recently proposed techniques, Hessian sketch and dual random projection, from a primal-dual point of view. We proposed accelerated methods for IHS and IDRP, from the preconditioned optimization perspective. By combining the primal and dual sketching techniques, we proposed a novel iterative primal-dual sketching approach, which substantially reduces the computational cost when solving sketched subproblems. We demonstrated applications of the iterative sketching techniques for distributed optimization when the data is partitioned by features or by samples. The proposed approach can be extended to solve more general problems. For example, by sketching the Newton step in a second-order optimization method, as done in [Pilanci and Wainwright, 2017], we will be able to solve regularized risk minimization problems with self-concordant losses.

In Chapter 5, we have developed several algorithms to achieve communication-computation balance in distributed optimization for empirical risk minimization. In particular, we present the first computational near-linear speed up in distributed optimization algorithms, even with large condition numbers.

In Chapter 6, we introduced and studied a shared-sparsity distributed multi-task learning problem. We presented a novel communication-efficient approach that required only one round of communication and achieves provable guarantees that compete with the centralized approach to leading order up to a generous bound on the number of machines. DSML can be easily extended to other types of structured sparsity, including sparse group lasso [Friedman et al., 2010], tree-guided group lasso [Kim and Xing, 2010] and the dirty model [Jalali et al., 2010].

In Chapter 7, we studied the problem of distributed representation learning for multiple tasks, discussed the implementation and guarantees for distributed convex optimization methods, and presented two novel algorithms to learn low-dimensional projection in a greedy way, which can be communication more efficient than distributed convex optimization approaches.

8.1 Future directions

In this section, we summarize several open questions related to the research in this thesis, and discuss future research directions.

For the sparse learning under homogeneous distributed setting, it is interesting to explore how to make EDSL provably work under even weaker assumptions. For example, EDSL requires $\mathcal{O}(s^2 \log p)$ samples per machine to match the centralized method in $\mathcal{O}(\log m)$ rounds of communications, however, it is not clear whether the sample size requirement can be improved, while still maintaining low-communication cost. Moreover, it is interesting to explore similar ideas to improve the computational cost of coupled distributed learning with shared sparsity [Wang et al., 2016b].

To understand the communication-computation trade-offs in distributed optimization better, there are several interesting questions to investigate in the future. First, are there algorithms that can match both the communication and computation lower bound in distributed optimization, or there is an inherent trade-off between communication and computation? Empirically, the `AMD-SVRG-P` algorithm appears to have optimal computational efficiency, with significantly improved communication efficiency over `AMD-SVRG`. It will be very interesting to establish theoretical guarantees for it. Second, if the inherent trade-offs exist, are there approaches that allow us to perform smooth transition between communication optimal algorithms and computational optimal algorithms?

When studying the distributed multi-task learning with shared sparsity, our main theoretical results were presented under the random sub-Gaussian design, however, the proofs are based on fixed design assumptions, namely Restricted Eigenvalue and Generalized Coherence conditions are imposed. These conditions are satisfied with high-probability under the random design. Furthermore, such conditions, or other similar conditions, are required for support recovery, but much weaker conditions are sufficient for obtaining low prediction error with the lasso or group lasso. An interesting open question is whether there exists a communication efficient method for distributed multi-task learning that requires sample complexity $n = O(s + (\log p)/m)$, like the group lasso, even without Restricted Eigenvalue and Generalized Coherence conditions, or whether beating the $n = O(s + \log p)$ sample complexity of the lasso in a more general setting inherently requires large amounts of communication.

As a long term research direction, it is interesting and important to obtain better theoretical understanding on algorithms that are often used in practice. For example, iteratively run several steps of stochastic gradient descent in parallel and average the iterates from multiple machines, have been widely used in practice (e.g. the federated learning system [McMahan et al., 2017]) as an important alternative of minibatch SGD, it would be interesting to theoretically prove its convergence, and understand when it outperforms minibatch SGD.

REFERENCES

Alekh Agarwal, Sahand Negahban, Martin J Wainwright, et al. Fast global convergence of gradient methods for high-dimensional statistical recovery. *The Annals of Statistics*, 40 (5):2452–2482, 2012.

Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

Ahmed Alaoui and Michael W Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems*, pages 775–783, 2015.

Zeyuan Allen-Zhu and Elad Hazan. Optimal black-box reductions between optimization objectives. In *Advances in Neural Information Processing Systems*, pages 1614–1622, 2016.

Yonatan Amit, Michael Fink, Nathan Srebro, and Shimon Ullman. Uncovering shared structures in multiclass classification. In *Proceedings of the 24th international conference on Machine learning*, pages 17–24. ACM, 2007.

Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in neural information processing systems*, pages 1756–1764, 2015.

Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging lapack's least-squares solver. *SIAM Journal on Scientific Computing*, 32(3):1217–1236, 2010.

Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1): 1–106, 2012.

Maria Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. In *Conference on Learning Theory*, pages 26–1, 2012.

Dror Baron, Marco F Duarte, Michael B Wakin, Shriram Sarvotham, and Richard G Baraniuk. Distributed compressive sensing. *arXiv preprint arXiv:0901.3403*, 2009.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Heather Battey, Jianqing Fan, Han Liu, Junwei Lu, and Ziwei Zhu. Distributed estimation and inference with statistical guarantees. *arXiv preprint arXiv:1509.05457*, 2015.

Ron Bekkerman, Mikhail Bilenko, and John Langford. *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

Aurélien Bellet, Yingyu Liang, Alireza Bagheri Garakani, Maria-Florina Balcan, and Fei Sha. A distributed frank-wolfe algorithm for communication-efficient sparse learning. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 478–486. SIAM, 2015.

Albert S Berahas, Raghu Bollapragada, Nitish Shirish Keskar, and Ermin Wei. Balancing communication and computation in distributed optimization. *arXiv preprint arXiv:1709.02999*, 2017.

Dimitri P Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical programming*, 129(2):163, 2011.

Dimitri P Bertsekas. Incremental aggregated proximal and augmented lagrangian algorithms. *arXiv preprint arXiv:1509.09257*, 2015.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997. ISBN 1886529019.

Peter J Bickel, Yaacov Ritov, Alexandre B Tsybakov, et al. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, 37(4):1705–1732, 2009.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

Olivier Bousquet and Léon Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

Christos Boutsidis and Alex Gittens. Improved matrix algorithms via the subsampled randomized hadamard transform. *SIAM Journal on Matrix Analysis and Applications*, 34(3): 1301–1340, 2013.

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

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

Mark Braverman, Ankit Garg, Tengyu Ma, Huy L Nguyen, and David P Woodruff. Communication lower bounds for statistical estimation problems via a distributed data processing inequality. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 1011–1020. ACM, 2016.

P. Bühlmann and S. Van De Geer. *Statistics for high-dimensional data: Methods, theory and applications*. Springer Science & Business Media, 2011.

Florentina Bunea et al. Honest variable selection in linear and logistic regression models via $\ell 1$ and $\ell 1+$ $\ell 2$ penalization. *Electronic Journal of Statistics*, 2:1153–1194, 2008.

Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Laurent Cavalier, GK Golubev, Dominique Picard, AB Tsybakov, et al. Oracle inequalities for inverse problems. *The Annals of Statistics*, 30(3):843–874, 2002.

Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Multi-task learning for boosting with application to web search ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1189–1198. ACM, 2010.

Li Cheng, Dale Schuurmans, Shaojun Wang, Terry Caelli, and Svn Vishwanathan. Implicit online learning with kernels. In *Advances in neural information processing systems*, pages 249–256, 2007.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *Advances in neural information processing systems*, pages 1647–1655, 2011.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM, 2010.

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Aaron Defazio. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems*, pages 676–684, 2016.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, 66(3):889–916, 2016.

Paramveer S Dhillon, Dean P Foster, Sham M Kakade, and Lyle H Ungar. A risk comparison of ordinary least squares vs ridge regression. *The Journal of Machine Learning Research*, 14(1):1505–1511, 2013.

Francesco Dinuzzo, Gianluigi Pillonetto, and Giuseppe De Nicolao. Client–server multitask learning from distributed datasets. *IEEE Transactions on Neural Networks*, 22(2):290–303, 2011.

David L Donoho and Xiaoming Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 2001.

Petros Drineas and Michael W Mahoney. Randnla: randomized numerical linear algebra. *Communications of the ACM*, 59(6):80–90, 2016.

Petros Drineas, Michael W Mahoney, S Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische mathematik*, 117(2):219–249, 2011.

Marco F Duarte and Yu Hen Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004.

John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2012.

John C Duchi, Michael I Jordan, Martin J Wainwright, and Yuchen Zhang. Optimality guarantees for distributed statistical estimation. *arXiv preprint arXiv:1405.0782*, 2014.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 3(1-2):95–110, 1956.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*, pages 2540–2548, 2015.

William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel computing*, 22(6):789–828, 1996.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

Bingsheng He and Xiaoming Yuan. On the o(1/n) convergence rate of the douglas–rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

Christina Heinze, Brian McWilliams, Nicolai Meinshausen, and Gabriel Krummenacher. Loco: Distributing ridge regression with random projections. *arXiv preprint arXiv:1406.3469*, 2014.

Christina Heinze, Brian McWilliams, and Nicolai Meinshausen. Dual-loco: Distributing statistical estimation using random projections. *arXiv preprint arXiv:1506.02554*, 2015.

Magnus R Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems'. *Journal of Research of the National Bureau of Standards*, 49(6), 1952.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, 162(1-2):165–199, 2017.

Daniel Hsu, Sham M Kakade, and Tong Zhang. Random design analysis of ridge regression. *Foundations of Computational Mathematics*, 14(3):569–600, 2014.

Cheng Huang and Xiaoming Huo. A distributed one-step estimator. *arXiv preprint arXiv:1511.01443*, 2015.

Martin Jaggi. Revisiting frank-wolfe: projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on International Conference on Machine Learning-Volume 28*, pages I–427. JMLR. org, 2013.

Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.

Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674. ACM, 2013.

Ali Jalali, Sujay Sanghavi, Chao Ruan, and Pradeep K Ravikumar. A dirty model for multi-task learning. In *Advances in neural information processing systems*, pages 964–972, 2010.

Adel Javanmard. Inference and estimation in high-dimensional data analysis. *PhD dissertation, Stanford University*, 2014.

Adel Javanmard and Andrea Montanari. Confidence intervals and hypothesis testing for high-dimensional regression. *Journal of Machine Learning Research*, 15:2869–2909, 2014.

Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th annual international conference on machine learning*, pages 457–464. ACM, 2009.

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

Michael I Jordan, Jason D Lee, and Yun Yang. Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, (just-accepted), 2018.

Rajiv Khanna, Ethan R Elenberg, Alexandros G Dimakis, Joydeep Ghosh, and Sahand Negahban. On approximation guarantees for greedy low rank optimization. In *International Conference on Machine Learning*, pages 1837–1846, 2017.

Seyoung Kim and Eric P Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 543–550. Omnipress, 2010.

Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Brian Kulis and Peter L. Bartlett. Implicit online learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 575–582, 2010.

Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015.

Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an o (1/t) convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012.

Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *Mathematical programming*, pages 1–49, 2017.

Maksim Lapin, Bernt Schiele, and Matthias Hein. Scalable multitask representation learning for scene classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1434–1441, 2014.

Jason D Lee, Qihang Lin, Tengyu Ma, and Tianbao Yang. Distributed stochastic variance reduced gradient methods by sampling extra data with replacement. *The Journal of Machine Learning Research*, 18(1):4404–4446, 2017a.

Jason D Lee, Qiang Liu, Yuekai Sun, and Jonathan E Taylor. Communication-efficient sparse regression. *Journal of Machine Learning Research*, 18(5):1–30, 2017b.

Peter J Lenk, Wayne S DeSarbo, Paul E Green, and Martin R Young. Hierarchical bayes conjoint analysis: recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science*, 15(2):173–191, 1996.

Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J. Smola. Efficient mini-batch training for stochastic optimization. pages 661–670.

Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation*, pages 583–598, 2014.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015.

Han Liu, Mark Palatucci, and Jian Zhang. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 649–656. ACM, 2009.

Sulin Liu, Sinno Jialin Pan, and Qirong Ho. Distributed multi-task relationship learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 937–946. ACM, 2017.

Karim Lounici, Massimiliano Pontil, Sara Van De Geer, and Alexandre Tsybakov. Oracle inequalities and optimal inference under group sparsity. *The Annals of Statistics*, 39(4): 2164–2204, 2011.

Junwei Lu, Guang Cheng, and Han Liu. Nonparametric heterogeneity testing for massive data. *arXiv preprint arXiv:1601.06212*, 2016.

Yichao Lu, Paramveer Dhillon, Dean P Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in neural information processing systems*, pages 369–377, 2013.

David G Luenberger. Introduction to linear and nonlinear programming. 1973.

Chenxin Ma, Virginia Smith, Martin Jaggi, Michael Jordan, Peter Richtarik, and Martin Takac. Adding vs. averaging in distributed primal-dual optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1973–1982, 2015.

Michael W Mahoney et al. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

Ali Makhdoumi and Asuman Ozdaglar. Convergence rate of distributed admm over networks. *IEEE Transactions on Automatic Control*, 62(10):5082–5095, 2017.

Richard P Martin, Amin M Vahdat, David E Culler, and Thomas E Anderson. Effects of communication latency, overhead, and bandwidth in a cluster architecture. In *Computer Architecture, 1997. Conference Proceedings. The 24th Annual International Symposium on*, pages 85–97. IEEE, 1997.

Peter McCullagh and John A Nelder. *Generalized Linear Models*, volume 37. CRC Press, 1989.

Ryan Mcdonald, Mehryar Mohri, Nathan Silberman, Dan Walker, and Gideon S Mann. Efficient large-scale distributed training of conditional maximum entropy models. In *Advances in Neural Information Processing Systems*, pages 1231–1239, 2009.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *The annals of statistics*, pages 1436–1462, 2006.

Nicolai Meinshausen and Bin Yu. Lasso-type recovery of sparse representations for high-dimensional data. *The Annals of Statistics*, pages 246–270, 2009.

Shahar Mendelson, Alain Pajor, and Nicole Tomczak-Jaegermann. Reconstruction and subgaussian operators in asymptotic geometric analysis. *Geometric and Functional Analysis*, 17(4):1248–1282, 2007.

Xiangrui Meng, Michael A Saunders, and Michael W Mahoney. Lsrn: A parallel iterative solver for strongly over-or underdetermined systems. *SIAM Journal on Scientific Computing*, 36(2):C95–C118, 2014.

Angelia Nedic, Asuman Ozdaglar, and Pablo A Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

Deanna Needell and Joel A Tropp. Paved with good intentions: analysis of a randomized block kaczmarz method. *Linear Algebra and its Applications*, 441:199–221, 2014.

Sahand N Negahban, Pradeep Ravikumar, Martin J Wainwright, and Bin Yu. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. *Statistical Science*, pages 538–557, 2012.

Arkadii Nemirovskii, David Borisovich Yudin, and Edgar Ronald Dawson. Problem complexity and method efficiency in optimization. 1983.

Yurii Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Atsushi Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems*, pages 1574–1582, 2014.

Jorge Nocedal and Stephen J Wright. *Numerical Optimization.* Springer, 2006.

Guillaume Obozinski, Ben Taskar, and Michael I Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2): 231–252, 2010.

Guillaume Obozinski, Martin J Wainwright, Michael I Jordan, et al. Support union recovery in high-dimensional multivariate regression. *The Annals of Statistics*, 39(1):1–47, 2011.

Alex Olshevsky and John N Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM review*, 53(4):747–772, 2011.

Samet Oymak and Joel A Tropp. Universality laws for randomized dimension reduction, with applications. *Information and Inference: A Journal of the IMA*, 2015.

Samet Oymak, Benjamin Recht, and Mahdi Soltanolkotabi. Isometric sketching of any set via the restricted isometry property. *arXiv preprint arXiv:1506.03521*, 2015.

Mert Pilanci and Martin J Wainwright. Randomized sketches of convex programs with sharp guarantees. *IEEE Transactions on Information Theory*, 61(9):5096–5115, 2015.

Mert Pilanci and Martin J Wainwright. Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares. *The Journal of Machine Learning Research*, 17(1):1842–1879, 2016.

Mert Pilanci and Martin J Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

Massimiliano Pontil and Andreas Maurer. Excess risk bounds for multitask learning with trace norm regularization. In *Conference on Learning Theory*, pages 55–76, 2013.

S Sundhar Ram, Angelia Nedić, and Venugopal V Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications*, 147(3):516–545, 2010.

Garvesh Raskutti and Michael Mahoney. Statistical and algorithmic perspectives on randomized sketching for ordinary least-squares. In *International Conference on Machine Learning*, pages 617–625, 2015.

Garvesh Raskutti, Martin J Wainwright, and Bin Yu. Restricted eigenvalue properties for correlated gaussian designs. *Journal of Machine Learning Research*, 11(Aug):2241–2259, 2010.

Pradeep Ravikumar, Martin J Wainwright, John D Lafferty, et al. High-dimensional ising model selection using ℓ1-regularized logistic regression. *The Annals of Statistics*, 38(3): 1287–1319, 2010.

Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(Dec):3413–3430, 2011.

Sashank J Reddi, Jakub Konečnỳ, Peter Richtárik, Barnabás Póczós, and Alex Smola. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105(36):13212–13217, 2008.

Jonathan D Rosenblatt and Boaz Nadler. On the optimality of averaging in distributed statistical learning. *Information and Inference: A Journal of the IMA*, 5(4):379–404, 2016.

Nicolas L Roux, Mark Schmidt, and Francis 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.

Mark Rudelson and Shuheng Zhou. Reconstruction from anisotropic random measurements. *IEEE Transactions on Information Theory*, 59(6):3434–3447, 2013.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Chris Sander and Reinhard Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins: Structure, Function, and Bioinformatics*, 9(1):56–68, 1991.

Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 143–152. IEEE, 2006.

Mark Schmidt, Nicolas L Roux, and Francis R Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in neural information processing systems*, pages 1458–1466, 2011.

Michael L Seltzer and Jasha Droppo. Multi-task learning in deep neural networks for improved phoneme recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6965–6969. IEEE, 2013.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

Shai Shalev-Shwartz and Nathan Srebro. Svm optimization: inverse dependence on training set size. In *Proceedings of the 25th international conference on Machine learning*, pages 928–935. ACM, 2008.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming: Series A and B*, 155(1-2):105–145, 2016.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Stochastic convex optimization. In *Conference on Learning Theory*, 2009.

Shai Shalev-Shwartz, Nathan Srebro, and Tong Zhang. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization*, 20(6): 2807–2832, 2010.

Shai Shalev-Shwartz, Alon Gonen, and Ohad Shamir. Large-scale convex minimization with a low-rank constraint. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 329–336. Omnipress, 2011a.

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011b.

Ohad Shamir. Without-replacement sampling for stochastic gradient methods. In *Advances in Neural Information Processing Systems*, pages 46–54, 2016.

Ohad Shamir and Nathan Srebro. On distributed stochastic optimization and learning. In *52nd Annual Allerton Conference on Communication, Control and Computing*, 2014.

Ohad Shamir, Nathan Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008, 2014.

Zuofeng Shang and Guang Cheng. Computational limits of a distributed algorithm for smoothing spline. *The Journal of Machine Learning Research*, 18(1):3809–3845, 2017.

Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

Brent Smith and Greg Linden. Two decades of recommender systems at amazon. com. *IEEE Internet Computing*, 21(3):12–18, 2017.

Virginia Smith, Simone Forte, Chenxin Ma, Martin Takac, Michael I Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *arXiv preprint arXiv:1611.02189*, 2016.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4427–4437, 2017.

Eleftherios Spyromitros-Xioufis, Grigorios Tsoumakas, William Groves, and Ioannis Vlahavas. Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, 104(1):55–98, 2016.

Karthik Sridharan, Shai Shalev-Shwartz, and Nathan Srebro. Fast rates for regularized objectives. In *Advances in Neural Information Processing Systems*, pages 1545–1552, 2009.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Panos Toulis and Edoardo M. Airoldi. Implicit stochastic gradient descent. *arXiv preprint arXiv:1408.2923*, 2014.

Joel A Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information theory*, 50(10):2231–2242, 2004.

Konstantinos Tsianos, Sean Lawlor, and Michael G Rabbat. Communication/computation tradeoffs in consensus-based distributed optimization. In *Advances in neural information processing systems*, pages 1943–1951, 2012.

John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

Berwin A Turlach, William N Venables, and Stephen J Wright. Simultaneous variable selection. *Technometrics*, 47(3):349–363, 2005.

Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):467–476, 2008.

Sara Van de Geer, Peter Bühlmann, Yaacov Ritov, Ruben Dezeure, et al. On asymptotically optimal confidence regions and tests for high-dimensional models. *The Annals of Statistics*, 42(3):1166–1202, 2014.

Sara A Van De Geer, Peter Bühlmann, et al. On the conditions used to prove oracle results for the lasso. *Electronic Journal of Statistics*, 3:1360–1392, 2009.

Sara A Van de Geer et al. High-dimensional generalized linear models and the lasso. *The Annals of Statistics*, 36(2):614–645, 2008.

Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized collaborative learning of personalized models over networks. *arXiv preprint arXiv:1610.05202*, 2016.

Vladimir Vapnik. *The nature of statistical learning theory.* Springer science & business media, 1995.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

Roman Vershynin. Estimation in high dimensions: a geometric perspective. In *Sampling theory, a renaissance*, pages 3–66. Springer, 2015.

Martin J Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming (lasso). *IEEE transactions on information theory*, 55(5):2183–2202, 2009.

Jialei Wang and Tong Zhang. Improved optimization of finite sums with minibatch stochastic variance reduced proximal iterations. *arXiv preprint arXiv:1706.07001*, 2017.

Jialei Wang, Mladen Kolar, and Nathan Srebro. Distributed multi-task learning with shared representation. *arXiv preprint arXiv:1603.02185*, 2016a.

Jialei Wang, Mladen Kolar, and Nathan Srerbo. Distributed multi-task learning. In *Artificial Intelligence and Statistics*, pages 751–760, 2016b.

Jialei Wang, Mladen Kolar, Nathan Srebro, and Tong Zhang. Efficient distributed learning with sparsity. In *International Conference on Machine Learning*, pages 3636–3645, 2017a.

Jialei Wang, Jason Lee, Mehrdad Mahdavi, Mladen Kolar, and Nati Srebro. Sketching meets random projection in the dual: A provable recovery algorithm for big and high-dimensional data. In *Artificial Intelligence and Statistics*, pages 1150–1158, 2017b.

Jialei Wang, Jason D Lee, Mehrdad Mahdavi, Mladen Kolar, Nathan Srebro, et al. Sketching meets random projection in the dual: A provable recovery algorithm for big and high-dimensional data. *Electronic Journal of Statistics*, 11(2):4896–4944, 2017c.

Jialei Wang, Weiran Wang, and Nathan Srebro. Memory and communication efficient distributed stochastic optimization with minibatch prox. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 1882–1919, 2017d.

Jialei Wang, Mladen Kolar, Nathan Srebro, and Tong Zhang. Communication-computation balanced optimization for distributed machine learning. In *Technical Report*, 2018.

Shusen Wang, Farbod Roosta-Khorasani, Peng Xu, and Michael W Mahoney. Giant: Globally improved approximate newton method for distributed optimization. *arXiv preprint arXiv:1709.03528*, 2017e.

Xiangyu Wang, David B Dunson, and Chenlei Leng. Decorrelated feature space partitioning for distributed sparse regression. In *Advances in Neural Information Processing Systems*, pages 802–810, 2016c.

Zheng Wang, Ming-Jun Lai, Zhaosong Lu, Wei Fan, Hasan Davulcu, and Jieping Ye. Orthogonal rank-one matrix pursuit for low rank matrix completion. *SIAM Journal on Scientific Computing*, 37(1):A488–A514, 2015.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.

David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

Blake E Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in neural information processing systems*, pages 3639–3647, 2016.

Tong Tong Wu, Yi Fang Chen, Trevor Hastie, Eric Sobel, and Kenneth Lange. Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics*, 25(6):714–721, 2009.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, 2007.

Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8(Jan): 35–63, 2007.

Jiyan Yang, Michael W Mahoney, Michael Saunders, and Yuekai Sun. Feature-distributed sparse regression: a screen-and-clean approach. In *Advances in Neural Information Processing Systems*, pages 2712–2720, 2016.

Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 629–637, 2013.

Tianbao Yang, Lijun Zhang, Rong Jin, and Shenghuo Zhu. Theory of dual-sparse regularized randomized reduction. In *International Conference on Machine Learning*, pages 305–314, 2015.

Yun Yang, Mert Pilanci, Martin J Wainwright, et al. Randomized sketches for kernels: Fast and optimal nonparametric regression. *The Annals of Statistics*, 45(3):991–1023, 2017.

Adams Wei Yu, Qihang Lin, and Tianbao Yang. Doubly stochastic primal-dual coordinate method for empirical risk minimization and bilinear saddle-point problem. *arXiv preprint arXiv:1508.03390*, 2015.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Ming Yuan and Yi Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

Ming Yuan, Ali Ekici, Zhaosong Lu, and Renato Monteiro. Dimension reduction and coefficient estimation in multivariate linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(3):329–346, 2007.

Cun-Hui Zhang and Stephanie S. Zhang. Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society: Series B*, 76 (1):217–242, Jul 2013.

Lijun Zhang, Mehrdad Mahdavi, and Rong Jin. Linear convergence with condition number independent access of full gradients. In *Advances in Neural Information Processing Systems*, pages 980–988, 2013a.

Lijun Zhang, Mehrdad Mahdavi, Rong Jin, Tianbao Yang, and Shenghuo Zhu. Recovering the optimal solution by dual random projection. In *Conference on Learning Theory*, pages 135–157, 2013b.

Lijun Zhang, Mehrdad Mahdavi, Rong Jin, Tianbao Yang, and Shenghuo Zhu. Random projections for classification: A recovery approach. *IEEE Transactions on Information Theory*, 60(11):7300–7316, 2014.

Tong Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17(9):2077–2098, 2005.

Yuchen Zhang and Lin Xiao. Disco: Distributed optimization for self-concordant empirical loss. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 362–370, 2015.

Yuchen Zhang and Lin Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *The Journal of Machine Learning Research*, 18(1):2939–2980, 2017.

Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.

Yuchen Zhang, John Duchi, Michael I Jordan, and Martin J Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems*, pages 2328–2336, 2013c.

Yuchen Zhang, John Duchi, and Martin Wainwright. Divide and conquer kernel ridge regression. In *Conference on Learning Theory*, pages 592–617, 2013d.

Yuchen Zhang, John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems*, pages 2328–2336, 2013e.

Yuchen Zhang, John C. Duchi, and Martin J. Wainwright. Communication-efficient algorithms for statistical optimization. *Journal of Machine Learning Research*, 14(1):3321–3363, 2013f.

Yuchen Zhang, John C. Duchi, and Martin J. Wainwright. Communication-efficient algorithms for statistical optimization. *Journal of Machine Learning Research*, pages 3321–3363, 2013g.

Yuchen Zhang, John Duchi, and Martin Wainwright. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *The Journal of Machine Learning Research*, 16(1):3299–3340, 2015.

Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pages 3468–3497, 2009.

Tianqi Zhao, Guang Cheng, and Han Liu. A partially linear framework for massive heterogeneous data. *arXiv preprint arXiv:1410.8570*, 2014a.

Tianqi Zhao, Mladen Kolar, and Han Liu. A general framework for robust testing and confidence regions in high-dimensional quantile regression. *arXiv preprint arXiv:1412.8724*, 2014b.

Shun Zheng, Jialei Wang, Fen Xia, Wei Xu, and Tong Zhang. A general distributed dual coordinate optimization framework for regularized loss minimization. *Journal of Machine Learning Research*, 18(115):1–52, 2017.

Jiayu Zhou, Jun Liu, Vaibhav A.Narayan, and Jieping Ye. Modeling disease progression via multi-task learning. *NeuroImage*, 78:233–248, 2013.

Ji Zhu and Trevor Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5(3):427–443, 2004.

Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.