New Optimization Methods for Modern Machine Learning

Sashank J. Reddi

November 2016

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Alexander J. Smola, Co-Chair
Barnabás Póczos, Co-chair
Geoffery J. Gordon
Suvrit Sra (Massachusetts Institute of Technology)
Stephen Boyd (Stanford University)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2016 Sashank J. Reddi
Keywords: machine learning, optimization, large-scale, distributed
Abstract

Modern Machine Learning (ML) systems pose several new statistical, scalability, privacy and ethical challenges. This thesis focuses on important challenges related to scalability, such as computational and communication efficiency, often encountered while solving ML problems in large-scale distributed settings.

The first part of the thesis investigates new optimization methods for solving large-scale nonconvex finite-sum problems, typically referred to as empirical risk minimization (ERM) in ML. Traditionally, most of the focus in ML has been on developing convex models and algorithms (e.g., SVM and logistic regression). However, recently, nonconvex models have surged into limelight (notably via deep learning) and have led to exciting progress — for instance these models have completely revolutionized areas like computer vision and natural language processing. But our understanding of optimization methods suitable for these problems is still very limited. Driven by this limitation, we develop and analyze a new line of optimization methods for nonconvex ERM problems aimed at resolving important open problems in the wider stochastic optimization literature. For example, we develop nonconvex stochastic methods that enjoy provably faster convergence (to stationary points) than SGD and its deterministic counterpart, gradient descent, thereby marking the first theoretical improvement in this line of research. We also discuss surprising challenges in nonsmooth and constrained nonconvex ERM problems and present a few preliminary results addressing them. Finally, we show that the key principles behind our methods are generalizable and can be used to overcome challenges in other important problems such as Bayesian posterior inference.

The second part of the thesis studies two critical aspects of modern distributed ML systems — asynchronicity and communication efficiency of optimization methods. In the context of asynchronous optimization, we study various asynchronous algorithms with fast convergence for finite-sum optimization problems and show that these methods achieve near linear speedups in sparse settings common to ML. In addition to asynchronicity, communication efficiency also plays an important role in determining the overall performance of the system. However, traditional optimization algorithms used in ML are often ill-suited for distributed environments with high communication cost. To address this issue, we discuss two different paradigms to achieve communication efficiency of algorithms in distributed environments and explore new algorithms with better communication complexity.
Chapter 1

Introduction

Machine learning (ML) and intelligent systems have become an indispensable part of our modern society. These systems are now used for variety of tasks that includes search engine, recommendation engines, self-driving cars and autonomous robots. Most of these systems rely on recognizing patterns in observable data in order to understand the data or make new predictions on the unseen data. The advent of modern data collection methods and increased computing machinery have fueled the development of such ML systems. However, modern ML applications also pose new challenges in terms of scalability and efficiency, privacy and ethics; thus, addressing them is critical to the development of the field. This thesis is a step in the direction of addressing these new challenges in modern ML applications.

We start our discussion by further explaining the goals of this thesis. Modern machine learning applications are heavily rooted in statistics and typically involve two major tasks: (i) constructing a model that generates the observable data. (ii) learn the parameters of the model using the observable data. This thesis particularly focuses on developing fast and efficient mathematical optimization methods to address problem (ii) in modern ML applications. For the purpose of our discussion, consider the classical problem of classification using a logistic regression classifier. The samples \( \{(z_i, y_i)\}_{i=1}^n \) where \( z_i \in \mathbb{R}^d \) and \( y_i \in \{-1, 1\} \) for all \( i \in [n] \) forms the dataset where \( z_i \) is referred to as features and \( y_i \) the corresponding class label. In this case, the optimization problem of our interest is:

\[
\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-y_i z_i^\top x)) + \frac{1}{2} \|x\|^2. \quad (1.1)
\]

The term \( \log(1 + \exp(-y_i z_i^\top x)) \) represents the loss with respect to the \( i^{th} \) sample. The term \( \|x\|^2 \), referred to as regularization, improves the quality of the solution by providing better generalization over unseen data. One of the most interesting aspect of this problem is that it is separable over sample data points. Statistically speaking, such an attribute results from the assumption that the sample points are i.i.d from a probability distribution. In modern ML applications, the number of data points \( n \) is very large, in which case exploiting the separable nature of the optimization problem becomes important. More generally, in most part of this thesis, we are
interested in solving optimization problems of the following form:

$$\min_{x \in \mathcal{X}} \frac{1}{n} \sum_{i=1}^{n} f_i(x) + r(x),$$

(1.2)

where $\mathcal{X}$ is a compact convex set. Optimization problems of this form, typically referred to as empirical risk minimization (ERM) problems or finite-sum problems, are central to most applications in ML. For example, in logistic regression problem, $f_i(x) = \log(1 + \exp(-y_iz_i^\top x))$, $r(x) = \frac{1}{2} \|x\|^2$ and $\mathcal{X} = \mathbb{R}^d$. In general, for supervised learning tasks, $x$, $f_i$ and $r$ represent the parameter of our interest, loss with respect to $i$th data point and regularization respectively. In most instances, a closed-form solution for problems of form (1.2) does not exist. Hence, one has to resort to numerical optimization techniques in order to obtain a solution. Numerical optimization methods based on first-order methods (i.e., based on gradient information of the function) are particularly favored in ML community due to their scalable nature. Popular methods include gradient descent, stochastic gradient descent and randomized coordinate descent. However, as will see later, these methods can be significantly improved by further exploiting the structure of the problem in (1.2).

Before proceeding any further, one needs to understand the characteristics and requirements of modern machine learning applications in order to appreciate the contributions of this work. Modern ML applications have added the following two new dimensions to the traditional ones.

1. **Increased complexity of the model.** Traditionally, most of the focus in machine learning has been on developing convex models and algorithms (e.g., SVM, logistic regression). However, recently, nonconvex models have surged into the limelight (notably via deep learning) and led to exciting progress – for instance these models have provided state-of-the-art performance and have completely revolutionized areas like computer vision, natural language processing. Thus, developing fast and principled optimization techniques for solving these complex models becomes important.

2. **Large-scale and distributed data.** With the advent of modern data collection methods, the size of the datasets used in ML applications have increased tremendously. Thus, the dataset is huge and distributed across several computing nodes. For example, large scale distributed machine learning systems such as the Parameter server [25], GraphLab [63] and TensorFlow [1] work with datasets sizes in the order of hundreds of terabytes. When dealing with datasets of such scale in distributed systems, computational and communication workloads need to be designed carefully.

The main focus of this thesis is to make the progress geared towards addressing these important aspects of the modern machine learning applications. Traditionally, for small-scale nonconvex optimization problems of form (1.2) that arise in ML, batch gradient methods have been used. However, in the large-scale setting i.e., $n$ is very large in (1.2), batch methods become intractable. In such scenarios, the popular stochastic gradient method (SGD) proposed by Robbins and Monro [54] is often preferred. SGD is an iterative first-order method wherein each step it takes a step in the negative direction of the stochastic approximation of the gradient. However, one of the fundamental issues with SGD is the noise due to the stochastic approximation of the gradient slows the convergence of the algorithm. In order to control the noise in the gradient, one has to typically decrease the step size as the algorithm proceeds. This in turn leads to slow
convergence and furthermore, raises the question of selection of step size and its decreasing rate. For the first part of the thesis, we address this problem in the context of nonconvex optimization problems by developing optimization algorithms with provable guarantees and faster convergence rates. Furthermore, we prove various interesting properties of these algorithms. These algorithms are based on the variance reduction techniques recently developed in the context of convex optimization [21, 56, 58]. Furthermore, we will investigate different scenarios including non-smoothness of the objective function and constrained minimization problems.

For the second part of my thesis, we look into the problem of asynchronous and distributed empirical risk minimization problems. In particular, we assume a setting where parallelism is important and communication between the nodes is expensive. In order to meet these requirements, the distributed optimization algorithm needs to be robust in terms of (i) synchronicity and (ii) the communication load of the overall system. To address the first part, we propose asynchronous stochastic algorithms with linear convergence guarantees when the objective function in (1.2) for strongly-convex. To our knowledge, this is the first work showing asynchronous system with linear convergence guarantees for strongly-convex objectives. These results also extend to the non-strongly convex case. More details of the results will be provided in Chapter 3. As mentioned earlier, another important constraint on the distributed optimization algorithm is the overall communication load on the system. To tackle this issue, we propose two different paradigms for minimizing the communication complexity of the distributed algorithm. We discuss and provide a few preliminary results on the performance of the algorithms in Chapter 3.

1.1 Overview of the Proposal Format

In this chapter, I provided a high level overview of my research that forms of the core of this thesis. In Chapters 2 and 3 I will discuss more details of the my research in addressing the two important issues raised in these chapters and provide details of the work that is planned to be incorporated in the thesis. Chapter 4 briefly summarizes my other research contributions that are not included in my thesis.

Approximate Timeline

- Date of Proposal: 16th November, 2016.
Chapter 2

Beyond Convexity: Fast Optimization Methods for Nonconvex Finite-sum Problems

In this chapter, we investigate fast stochastic methods for non-convex finite-sum problems. In particular, we study nonconvex finite-sum problems of the form

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

(2.1)

where neither $f$ nor the individual $f_i$ ($i \in [n]$) are necessarily convex; just Lipschitz smooth (i.e., Lipschitz continuous gradients). Problems of this form arise naturally in ML in the form of empirical risk minimization (ERM). We use $\mathcal{F}_n$ to denote all functions of the form (2.1). We optimize such functions in the Incremental First-order Oracle (IFO) framework defined below.

Definition 2.0.1 For $f \in \mathcal{F}_n$, an IFO takes an index $i \in [n]$ and a point $x \in \mathbb{R}^d$, and returns the pair $(f_i(x), \nabla f_i(x))$.

IFO based complexity analysis was introduced to study lower bounds for finite-sum problems. Algorithms that use IFOs are favored in large-scale applications as they require only a small amount first-order information at each iteration. Two fundamental models in machine learning that profit from IFO algorithms are (i) empirical risk minimization, which typically uses convex finite-sum models; and (ii) deep learning, which uses nonconvex ones.

The prototypical IFO algorithm, stochastic gradient descent (SGD\textsuperscript{1}) has witnessed tremendous progress in the recent years. By now a variety of accelerated, parallel, and faster converging versions are known. Among these, of particular importance are variance reduced (VR) stochastic methods [9, 21, 56], which have delivered exciting progress such as linear convergence rates (for strongly convex functions) as opposed to sublinear rates of ordinary SGD [35, 54]. Similar (but not same) benefits of VR methods can also be seen in smooth convex functions. The SVRG algo-

\textsuperscript{1}We use ‘incremental gradient’ and ‘stochastic gradient’ interchangeably, though we are only interested in finite-sum problems.

5
Table 2.1: Table comparing the best IFO complexity of different algorithms discussed in this chapter. The complexity is measured in terms of the number of oracle calls required to achieve an $\epsilon$-accurate solution (see Definition 2.1.1). Here, by fixed step size, we mean that the step size of the algorithm is fixed and does not dependent on $\epsilon$ (or alternatively $T$, the total number of iterations). The complexity of gradient dominated functions refers to the number of IFO calls required to obtain $\epsilon$-accurate solution for a $\tau$-gradient dominated function (see Section 2.1 for the definition).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Nonconvex</th>
<th>Convex</th>
<th>Gradient Dominated</th>
<th>Fixed Step Size?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>$O\left(\frac{1}{\epsilon^2}\right)$</td>
<td>$O\left(\frac{1}{\epsilon^2}\right)$</td>
<td>$O\left(\frac{1}{\epsilon^2}\right)$</td>
<td>×</td>
</tr>
<tr>
<td>GradientDescent</td>
<td>$O\left(\frac{n}{\epsilon}\right)$</td>
<td>$O\left(\frac{n}{\epsilon}\right)$</td>
<td>$O\left(n\tau \log(1/\epsilon)\right)$</td>
<td>✓</td>
</tr>
<tr>
<td>SVRG</td>
<td>$O\left(n + \frac{n^2/3}{\epsilon}\right)$</td>
<td>$O\left(n + \sqrt{n}/\epsilon\right)$</td>
<td>$O\left((n + n^2/3\tau) \log(1/\epsilon)\right)$</td>
<td>✓</td>
</tr>
</tbody>
</table>

Algorithm of [21] is particularly attractive here because of its low storage requirement in comparison to the algorithms in [9, 56].

Despite the meteoric rise of VR methods, their analysis for general nonconvex problems is largely missing. [21] remark on convergence of SVRG when $f \in \mathcal{F}_n$ is locally strongly convex and provide compelling experimental results (Fig. 4 in [21]). However, problems encountered in practice are typically not even locally convex, let alone strongly convex. The current analysis of SVRG does not extend to nonconvex functions as it relies heavily on convexity for controlling the variance. Given the dominance of stochastic gradient methods in optimizing deep neural nets and other large nonconvex models, theoretical investigation of faster nonconvex stochastic methods is much needed.

Convex VR methods are known to enjoy the faster convergence rate of GradientDescent but with a much weaker dependence on $n$, without compromising the rate like SGD. However, it is not clear if these benefits carry beyond convex problems, prompting the central question of this chapter:

*For nonconvex functions in $\mathcal{F}_n$, can one achieve convergence rates faster than both SGD and GradientDescent using an IFO? If so, then how does the rate depend on $n$ and on the number of iterations performed by the algorithm?*

Perhaps surprisingly, we provide an affirmative answer to this question by showing that a careful selection of parameters in SVRG leads to faster convergence than both GradientDescent and SGD. To our knowledge, ours is the first work to improve convergence rates of SGD and GradientDescent for IFO-based nonconvex optimization. The key complexity results are listed in Table 2.1

### 2.1 Background

We say $f$ is $L$-smooth if there is a constant $L$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall \ x, y \in \mathbb{R}^d.$$  

Throughout, we assume that the functions $f_i$ in (2.1) are $L$-smooth, so that $\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$ for all $i \in [n]$. Such an assumption is very common in the analysis of first-order...
methods. A function \( f \) is called \( \lambda \)-strongly convex if there is \( \lambda \geq 0 \) such that

\[
f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \frac{\lambda}{2} \|x - y\|^2 \quad \forall x, y \in \mathbb{R}^d.
\]

The quantity \( \kappa := \frac{L}{\lambda} \) is called the condition number of \( f \), whenever \( f \) is \( L \)-smooth and \( \lambda \)-strongly convex. We say \( f \) is non-strongly convex when \( f \) is \( 0 \)-strongly convex.

We also recall the class of gradient dominated functions \([39, 42]\), where a function \( f \) is called \( \tau \)-gradient dominated if for any \( x \in \mathbb{R}^d \)

\[
f(x) - f(x^*) \leq \tau \|\nabla f(x)\|^2,
\]

where \( x^* \) is a global minimizer of \( f \). Note that such a function \( f \) need not be convex; it is also easy to show that a \( \lambda \)-strongly convex function is \( \frac{1}{2\lambda} \)-gradient dominated.

We analyze convergence rates for the above classes of functions. Following \([16, 38]\) we use \( \|\nabla f(x)\|^2 \leq \epsilon \) to judge when is iterate \( x \) approximately stationary. Contrast this with SGD for convex \( f \), where one uses \( |f(x) - f(x^*)| \) or \( \|x - x^*\|^2 \) as a convergence criterion. Unfortunately, such criteria cannot be used for nonconvex functions due to the hardness of the problem. For our analysis, we need the following definition.

**Definition 2.1.1** A point \( x \) is called \( \epsilon \)-accurate if \( \|\nabla f(x)\|^2 \leq \epsilon \). A stochastic iterative algorithm is said to achieve \( \epsilon \)-accuracy in \( t \) iterations if \( \mathbb{E}[\|\nabla f(x^t)\|^2] \leq \epsilon \), where the expectation is over the stochasticity of the algorithm.

We measure the efficiency of the algorithms in terms of the number of IFO calls made by the algorithm (IFO complexity) to achieve an \( \epsilon \)-accurate solution. Throughout this chapter, we hide the dependence of IFO complexity on Lipschitz constant \( L \), and the initial point (in terms of \( \|x^0 - x^*\|^2 \) and \( f(x^0) - f(x^*) \)) for a clean comparison. We introduce one more definition useful in the analysis of SGD methods for bounding the variance.

**Definition 2.1.2** We say a function \( f \in \mathcal{F}_n^L \) has a \( \sigma \)-bounded if \( \|\nabla f_i(x) - \nabla f(x)\| \leq \sigma \) for all \( i \in [n] \) and \( x \in \mathbb{R}^d \).

### 2.2 Algorithms

Stochastic gradient descent (SGD) is one of the simplest iterative algorithms for solving \((2.1)\). The update of SGD is of the following form:

\[
x^{t+1} = x^t - \eta_t \left[ \frac{1}{b} \sum_{i \in I_t} \nabla f_i(x^t) \right],
\]

where \( I_t \) is a set chosen uniformly randomly (with replacement) from \([n]\) such that \( |I_t| = b \). SGD uses an unbiased estimate of the gradient at each iteration. In particular, \( \mathbb{E}[\frac{1}{b} \sum_{i \in I_t} \nabla f_i(x)] = \nabla f(x^t) \). Under appropriate conditions, \([16]\) establish convergence rate of SGD to a stationary point of \( f \). Their results include the following theorem.

**Corollary 2.2.0.1 (Informal)** Suppose function \( f \) has \( \sigma \)-bounded gradient, then the IFO complexity of SGD to obtain an \( \epsilon \)-accurate solution is \( O(1/\epsilon^2) \) under appropriate selection of step size and \( b = 1 \).
Algorithm 1: SVRG($x^0, T, m, \{p_i\}_{i=0}^m, b$)

1: **Input:** $\tilde{x}^0 = x^0, x^0 \in \mathbb{R}^d$, epoch length $m$, step sizes $\{\eta_i > 0\}_{i=0}^{m-1}$, $S = \lceil T/m \rceil$, discrete probability distribution $\{p_i\}_{i=0}^m$, mini-batch size $b$

2: **for** $s = 0$ **to** $S - 1$ **do**

3: $x^0_{s+1} = x^0_m$

4: $g^{s+1} = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{x}^s)$

5: **for** $t = 0$ **to** $m - 1$ **do**

6: Choose a mini-batch (uniformly random with replacement) $I_t \subset [n]$ of size $b$

7: $v^{s+1}_t = \frac{1}{b} \sum_{i \in I_t} (\nabla f_i(x^{s+1}_t) - \nabla f_i(\tilde{x}^s)) + g^{s+1}$

8: $x^{s+1}_{t+1} = x^{s+1}_t - \eta_t v^{s+1}_t$

9: **end for**

10: $\tilde{x}^{s+1} = \sum_{i=0}^{m} p_i x^{s+1}_i$

11: **end for**

12: **Output:** Iterate $x_a$ chosen uniformly random from $\{x^{s+1}_t \}_{t=0}^{m-1}$.

As seen in Corollary 2.2.0.1, SGD has a convergence rate of $O(1/\sqrt{T})$. This rate is not improvable in general even when the function is (non-strongly) convex in the pure stochastic setting [36]. This barrier is due to the variance introduced by the stochasticity of the gradients, and it is not clear if better rates can be obtained SGD even for (non-strongly) convex $f \in F_n$.

2.2.1 Nonconvex SVRG Algorithm

We now turn our focus to variance reduced methods. We first investigate a variant of SVRG [21], an algorithm recently shown to be very effective for reducing variance in convex problems. It has gained considerable interest in both machine learning and optimization communities. We seek to understand its benefits for nonconvex optimization. For reference, Algorithm 1 presents SVRG’s pseudocode.

Observe that Algorithm 1 operates in epochs. At the end of epoch $s$, a full gradient is calculated at the point $\tilde{x}^s$, requiring $n$ calls to the IFO. Within its inner loop SVRG performs $m$ stochastic updates. The total number of IFO calls for each epoch is thus $\Theta(m + n)$. For $m = 1$, the algorithm reduces to the classic GRADIENT DESCENT algorithm. Suppose $m$ is chosen to be $O(n)$ (typically used in practice), then the total IFO calls per epoch is $\Theta(n)$. To enable a fair comparison with SGD, we assume that the total number of inner iterations across all epochs in Algorithm 1 is $T$. Also note a simple but important implementation detail: as written, Algorithm 1 requires storing all the iterates $x^{s+1}_t (0 \leq t \leq m)$. This storage can be avoided by keeping a running average with respect to the probability distribution $\{p_i\}_{i=0}^m$.

Algorithm 1 attains linear convergence for strongly convex $f$ [21]: for non-strongly convex functions, rates faster than SGD can be shown by using an indirect perturbation argument—see e.g., [22, 62].

We prove the following result for SVRG algorithm in the context of non-convex finite sum minimization problems.
Theorem 2.2.1 Suppose \( b \leq n \) in Algorithm 1. Let \( p_i = 0 \) for \( 0 \leq i < m \) and \( p_m = 1 \), \( T \) be a multiple of \( m \) and \( \eta_t = \eta = \rho/L \) for all \( 0 \leq t \leq m - 1 \), where \( 0 < \rho \leq \min\{\frac{1}{4}, \sqrt{\frac{b}{3m}}\} \) and satisfies the condition \( 144m^2\rho^3 \leq b + 6m\rho^2 \). Then for the output \( x_a \) of Algorithm 1 with mini-batch size \( b \), we have the following:

\[
\mathbb{E}[\|\nabla f(x_a)\|^2] \leq \frac{8L[f(x^0) - f(x^*)]}{\rho T},
\]

where \( x^* \) is optimal for (2.1).

Theorem 2.2.1 provides a general convergence result for SVRG algorithm. The following result for specific parameter setting, follows as a simple corollary of Theorem 2.2.1.

Corollary 2.2.1.1 Suppose \( b \leq n^\alpha \) for some \( 0 \leq \alpha \leq 1 \) and \( m = \lfloor \frac{n^{3\alpha/2}}{2b} \rfloor \). Let \( p_i = 0 \) for \( 0 \leq i < m \) and \( p_m = 1 \), \( T \) be a multiple of \( m \) and \( \eta_t = \eta = \frac{b}{4Lm^\alpha} \) for all \( 0 \leq t \leq m - 1 \). Then for the output \( x_a \) of Algorithm 1 with mini-batch size \( b \), we have the following:

\[
\mathbb{E}[\|\nabla f(x_a)\|^2] \leq \frac{32n^\alpha L[f(x^0) - f(x^*)]}{bT},
\]

where \( x^* \) is optimal for (2.1).

We would like to add few remarks for Corollary 2.2.1.1. The above result shows an interesting interplay between the step size and the IFO complexity of SVRG algorithm. When \( \alpha \leq 2/3 \) in Corollary 2.2.1.1 (larger step size), the IFO complexity is dominated by the cost of computing the average gradient at the end of the epoch. On the other hand, when \( \alpha > 2/3 \) (smaller step size), the convergence rate slows down, as evident from the convergence rate in Corollary 2.2.1.1. The optimal choice of step size with respect to the bounds obtained in this chapter is for \( \alpha = 2/3 \) in Corollary 2.2.1.1 which provides the main result of this chapter. Note that in this setting, the convergence rate of SVRG algorithm is better than that of GRADIENT DESCENT by a factor \( n^{1/3} \).

A natural question arises if we these convergence rates can be further improved for a special class of non-convex optimization problems. We provide an affirmative answer to this question. In particular, for the class of gradient dominated functions (see (2.2)), we prove the following convergence rate to a global minima.

Corollary 2.2.1.2 (Informal) If \( f \in F_n \) is \( \tau \)-gradient dominated, the IFO complexity of a variant of Alg. 1 (under certain parameter settings) to compute an \( \epsilon \)-accurate solution is \( \tilde{O}((n + \tau n^{2/3}) \log(1/\epsilon)) \).

### 2.3 Incremental Method for Nonconvex Optimization

In the previous section, we investigated SVRG for solving (2.1). Note that SVRG is not a fully “incremental” algorithm since it requires calculation of the full gradient once per epoch. An alternative to SVRG is the algorithm proposed in [9] (popularly referred to as SAGA). We build upon the work of [9] to develop a nonconvex variant of SAGA.

Algorithm 2 presents pseudocode for SAGA. Note that the update \( v^t \) (Line 5) is unbiased, i.e., \( \mathbb{E}[v^t] = \nabla f(x^t) \). This is due to the uniform random selection of index \( i_t \). It can be seen
Algorithm 2: SAGA($x^0, T, \eta$)

1: **Input:** $x^0 \in \mathbb{R}^d$, $\alpha^0_i = x^0$ for $i \in [n]$, number of iterations $T$, step size $\eta > 0$
2: $g^0 = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\alpha^0_i)$
3: for $t = 0$ to $T - 1$
4: Uniformly randomly pick $i_t, j_t$ from $[n]$
5: $v^t = \nabla f_{i_t}(x^t) - \nabla f_{i_t}(\alpha^t_{i_t}) + g^t$
6: $x^{t+1} = x^t - \eta v^t$
7: $\alpha^{t+1}_{j_t} = x^{t}$ and $\alpha^{t+1}_{j_t} = \alpha^{t}_{j_t}$ for $j \neq j_t$
8: $g^{t+1} = g^t - \frac{1}{n} (\nabla f_{j_t}(\alpha^{t}_{j_t}) - \nabla f_{j_t}(\alpha^{t+1}_{j_t}))$
9: end for
10: **Output:** Iterate $x_a$ chosen uniformly random from $\{x^t\}_{t=0}^{T-1}$.

in Algorithm 2 that SAGA maintains gradients at $\alpha_i$ for $i \in [n]$. This additional set is critical to reducing the variance of the update $v^t$. At each iteration of the algorithm, one of the $\alpha_i$ is updated to the current iterate. An implementation of SAGA requires storage and updating of gradients $\nabla f_i(\alpha_i)$; the storage cost of the algorithm is $nd$. While this leads to higher storage in comparison to SGD, this cost is often reasonable for many applications. Furthermore, this cost can be reduced in the case of specific models; refer to [9] for more details.

**Theorem 2.3.1 (Informal)** Suppose $f \in \mathcal{F}_n$. For the step size $\eta = 1/(3Ln^{2/3})$, Algorithm 2 has the following convergence rate:

$$
\mathbb{E}[\| \nabla f(x_a) \|^2] \leq \frac{4n^{2/3}[f(x^0) - f(x^*)]}{T},
$$

where $x^*$ is an optimal solution to the problem in (2.1) and $x_a$ is the output of Algorithm 2.

The convergence rate of SAGA is similar to that of SVRG. Furthermore, a variant of SAGA can be shown to have linear convergence to the global minimum. Furthermore, SAGA also enjoys all the benefits of the GRADIENT DESCENT, albeit with a much weaker dependence on $n$. We refer interested readers to [51] for more details about the algorithm.

### 2.4 Proposed Work

In the previous sections, we have investigated fast stochastic methods for *unconstrained* and *smooth* non-convex finite-sum minimization problems. In this section, we discuss the setting where the objective function is non-smooth. More specifically, we are interested in the following optimization problem:

$$
\min_{x \in \mathcal{X}} F(x) := f(x) + h(x), \quad \text{where } f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x), \quad \text{(2.3)}
$$

and each $f_i : \mathbb{R}^d \to \mathbb{R}$ is smooth (possibly nonconvex) for all $i \in \{1, \ldots, n\} \triangleq [n]$, while $h : \mathbb{R}^d \to \mathbb{R}$ is nonsmooth but convex and relatively simple.
**Proximal Nonconvex Finite-sum problems.** A popular approach to handle non-smoothness in convex problems is via proximal operators \([30, 55]\), but as we will soon see, this approach does not work so easily for the nonconvex problem \((2.1)\). Nevertheless, recall that proper closed convex function \(h\), the proximal operator is defined as
\[
\text{prox}_{\eta h}(x) := \arg\min_{y \in X} \left( h(y) + \frac{1}{2\eta} \|y - x\|^2 \right), \quad \text{for } \eta > 0. \tag{2.4}
\]

The power of proximal operators lies in how they generalize projections: e.g., if \(h\) is the indicator function \(I_C(x)\) of a closed convex set \(C\), then \(\text{prox}_{I_C}(x) \equiv \text{proj}_C(x) \equiv \arg\min_{y \in C} \|y - x\|\).

Proximal methods are particularly useful when the proximal operator of \(h\) is easy to compute. This is true for many applications in machine learning and statistics including \(\ell_1\) regularization, box-constraints, simplex constraints, among others \([5, 41]\). For this discussion, assume an access to a proximal oracle (PO) that takes a point \(x \in \mathbb{R}^d\) and returns the output of (2.4).

A practical approach for solving problems of form \((2.3)\) is by proximal stochastic gradient (PROXSGD), which performs the iteration
\[
x_{t+1} = \text{prox}_{\eta_t h} \left( x_t - \frac{\eta_t}{|I_t|} \sum_{i \in I_t} \nabla f_i(x_t) \right), \quad t = 0, 1, \ldots, \tag{2.5}
\]

where \(I_t\) (referred to as minibatch) is a randomly chosen set (with replacement) from \([n]\) and \(\eta_t\) is a step size. Non-asymptotic convergence of PROXSGD was also shown recently, as noted below.

**Theorem 2.4.1 (Informal) [17]:** The number of IFO and PO calls made by PROXSGD, i.e., iteration \((2.5)\), to reach \(\epsilon\) close to a stationary point is \(O(1/\epsilon^2)\) and \(O(1/\epsilon)\) respectively. For achieving this convergence, we impose batch sizes \(|I_t|\) that increase and step sizes \(\eta_t\) that decrease with \(1/\epsilon\).

Notice that the PO complexity of PROXSGD is similar to GRADIENTDESCENT, but its IFO complexity is independent of \(n\); though, this benefit comes at the cost of an extra \(1/\epsilon^2\) factor. Furthermore, the step size must decrease with \(1/\epsilon\) (or alternatively decay with the number of iterations of the algorithm). The same two aspects are also seen for convex stochastic gradient, in both the smooth and proximal versions. However, in the nonconvex setting there is a key third and more important aspect: the minibatch size \(|I_t|\) increases with \(1/\epsilon\). To understand this aspect, consider the case where \(|I_t|\) is a constant (independent of both \(n\) and \(\epsilon\)), typically the choice used in practice. In this case, the above convergence result no longer holds and it is **not** clear if PROXSGD even converges to a stationary point at all! To clarify, a decreasing step size \(\eta_t\) trivially ensures convergence as \(t \to \infty\), but the limiting point is not necessarily stationary. On the other hand, increasing \(|I_t|\) with \(1/\epsilon\) can easily lead to \(|I_t| \geq n\) for reasonably small \(\epsilon\), which effectively reduces the algorithm to (batch) GRADIENTDESCENT. This problem does not afflict smooth nonconvex problems \((h \equiv 0)\), where convergence with constant minibatches is known \([16, 50, 51]\). Thus, there is a fundamental gap in our understanding of stochastic methods for nonsmooth nonconvex problems. Given the ubiquity of nonconvex models in machine learning, bridging this gap is important. We do so by analyzing stochastic proximal methods with guaranteed convergence for constant minibatches, and faster convergence with minibatches independent of \(1/\epsilon\). We have preliminary theoretical and practical results showing convergence
of proximal versions of SVRG and SAGA for constant minibatch size. I plan to further investigate the convergence behavior of these fast stochastic methods.

**Nonconvex Frank-Wolfe Variants.** Consider the following special case of (2.3):

$$\min_{x \in \mathcal{X}} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

We assume that \( f \), and \( f_i \ (i \in [n]) \) are all differentiable, but possibly nonconvex; the domain \( \mathcal{X} \) is convex and compact. In this case, the proximal operator reduces to a projection operator and can hence, be solved using the special case of PROXSGD — projected stochastic gradient descent. In many real settings, the cost projecting onto \( \Omega \) can be very high (e.g., projecting onto the trace-norm ball, onto base polytopes in submodular minimization [15]); and in extreme cases projection can even be computationally intractable [7].

In such cases, projection based methods like projected stochastic gradient descent become impractical. This difficulty underlies the recent surge of interest in Frank-Wolfe methods [14, 19] (also known as conditional gradient), due to their projection-free property. In particular, FW methods avoid the expensive projection operation and requires just a linear oracle that solves problems of the form \( \min_{x \in \Omega} \langle x, g \rangle \) at each iteration. Despite the remarkable success of FW approaches in the convex setting, including stochastic problems [18], their applicability and non-asymptotic convergence for nonconvex optimization is largely unstudied. Given the vast importance of nonconvex models in machine learning (e.g., in deep learning) and the need to incorporate non-trivial constraints in such models, it is imperative to develop scalable, projection-free methods. I plan to develop new FW methods towards this goal. We already have few preliminary results demonstrating the power of using variance reduced approximation of the gradient in the context of frank-wolfe methods for the nonconvex finite-sum optimization problem of our interest.

**Convergence to local minimizers & Lower Bounds for nonconvex optimization** All the results discussed so far show convergence to a stationary point of the nonconvex optimization problem. While convergence to a global minimizer is in general NP-hard for nonconvex optimization and hence, unlikely in the general setting, a natural question that arises is whether we can at least show stronger property of convergence to local minimizers. Recently, GRADIENT DESCENT with random initialization has been shown to asymptotically converge to local minimizers [24]. It will be interesting to show similar convergence properties for the algorithms discussed in this chapter. I plan to pursue this direction as part of my thesis work. We have very preliminary results showing a variants of the algorithms presented here can indeed converge to local minimizers.

Another interesting direction I plan to pursue is to derive the lower bounds for the optimization problems of form [2.1]. The convergence rate of both SVRG and SAGA, as mentioned earlier in the chapter, is surprisingly, \( O(n + n^{2/3}/\epsilon) \) in both the cases. An important question is whether these rates are tight and if not, can these rates can be further improved by either providing a tighter analysis or using a different algorithm altogether.

**Stochastic MCMC.** While so far we have discussed the benefits of using variance reduction techniques in the setting of finite-sum optimization, we show that these principles can also be used to improve other important algorithms in machine learning. We are particularly interested in
the problem of Bayesian posterior inference using stochastic MCMC methods. Gradient-based Monte Carlo methods such as Langevin Dynamics and Hamiltonian Monte Carlo [31] allow us to use gradient information to efficiently explore posterior distributions over continuous-valued parameters. By traversing contours of a potential energy function based on the posterior distribution, these methods allow us to make large moves in the sample space. Although gradient-based methods are efficient in exploring the posterior distribution, they are limited by the computational cost of computing the gradient and evaluating the likelihood on large datasets. As a result, stochastic variants are a popular choice when working with large data sets [61]. As a first step, in a recent NIPS paper [52], we show that by incorporating variance reduced approximation of the gradient in stochastic gradient langevin dynamics (SGLD) — a special case of stochastic MCMC method — one can ensure faster mixing than SGLD. I plan to further explore this direction in order to obtain faster variants of other stochastic MCMC methods.

2.4.1 Experiments

As part of my thesis, I plan to implement the proposed algorithms on a few large scale nonconvex ERM problems. I am, especially, interested in investigating the empirical performance of the algorithms in training deep learning models such as deep feedforward neural networks and deep autoencoders.
Chapter 3

Large-Scale Optimization: Asynchronous and Communication Efficient Optimization for Machine Learning

In this chapter, we investigate asynchronous and communication efficient optimization methods for problems that typically arise in machine learning. For the first part of the chapter, we restrict ourselves to asynchronous optimization methods. Handling asynchronicity in optimization algorithms is important to modern ML because systems based on synchronous algorithms are often slow and do not exploit parallelism in distributed systems. We start our discussion by revisiting the following finite-sum formulation in the previous section:

$$\min_{x \in \mathbb{R}^d} f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x).$$

(3.1)

For most part of this chapter, we assume that the functions $f_i$ are convex. Although the value of VR methods for solving optimization problems of form (3.1) have great value in general, for large-scale problems we still require parallel or distributed processing. And in this setting, asynchronous variants of SGD remain indispensable [3, 11, 26, 44, 59, 65]. Therefore, a key question is how to develop asynchronous and distributed variants of finite-sum VR algorithms. We answer one part of this question by developing new asynchronous parallel stochastic gradient methods that provably converge at a linear rate for smooth strongly convex finite-sum problems.

We make the following two core contributions: (i) a formal general framework for variance reduced stochastic methods based on discussions in [9]; and (ii) asynchronous parallel VR algorithms within this framework. Our general framework presents a formal unifying view of several VR methods (e.g., it includes SAGA and SVRG as special cases) while expressing key algorithmic and practical tradeoffs concisely. Thus, it yields a broader understanding of VR methods, which helps us obtain asynchronous parallel variants of VR methods. Under sparse-data settings common to machine learning problems, our parallel algorithms attain speedups that scale near linearly with the number of processors.

Related work. As already mentioned, our work is closest to (and generalizes) SAG [56], SAGA [9], SVRG [21] and S2GD [22], which are primal methods. Also closely related are dual
Algorithm 3: Generic Stochastic Variance Reduction Algorithm

Data: \( x^0 \in \mathbb{R}^d, \alpha^0_i = x^0 \forall i \in [n] \triangleq \{1, \ldots, n\} \), step size \( \eta > 0 \)

Randomly pick a \( I_T = \{i_0, \ldots, i_T\} \) where \( i_t \in \{1, \ldots, n\} \forall t \in \{0, \ldots, T\} \);

for \( t = 0 \) to \( T \) do

Update iterate as \( x^{t+1} \leftarrow x^t - \eta (\nabla f_{i_t}(x^t) - \nabla f_{i_t}(\alpha^t_i) + \frac{1}{n} \sum_i \nabla f_i(\alpha^t_i)) \);

\( A^{t+1} = \text{SCHEDULEUPDATE}(\{x^t\}_{t=0}^{t+1}, A^t, t, I_T) \);

end

return \( x^T \)

methods such as SDCA \[58\] and Finito \[10\], and in its convex incarnation MISO \[29\]; a more precise relation between these dual methods and VR stochastic methods is described in Defazio’s thesis \[8\]. By their algorithmic structure, these VR methods trace back to classical non-stochastic incremental gradient algorithms \[6\], but by now it is well-recognized that randomization helps obtain much sharper convergence results (in expectation). Proximal \[62\] and accelerated VR methods have also been proposed \[40, 57\]; we leave a study of such variants of our framework as future work. Finally, there is recent work on lower-bounds for finite-sum problems \[2\].

Within asynchronous SGD algorithms, both parallel \[44\] and distributed \[3, 34\] variants are known. In this chapter, we focus our attention on the parallel setting. A different line of methods is that of (primal) coordinate descent methods, and their parallel and distributed variants \[27, 28, 37, 45, 53\]. Our asynchronous methods share some structural assumptions with these methods. Finally, the recent work \[23\] generalizes S2GD to the mini-batch setting, thereby also permitting parallel processing, albeit with more synchronization and allowing only small mini-batches.

3.1 A General Framework for VR Stochastic Methods

We focus on instances of (2.1) where the cost function \( f(x) \) has an \( L \)-Lipschitz gradient, so that

\[ \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \]

and it is \( \lambda \)-strongly convex, i.e., for all \( x, y \in \mathbb{R}^d \),

\[ f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \frac{\lambda}{2}\|x - y\|^2. \]  \hspace{1cm} (3.2)

While our analysis focuses on strongly convex functions, we can extend it to just smooth convex functions along the lines of \[9, 62\].

Inspired by the discussion on a general view of variance reduced techniques in \[9\], we now describe a formal general framework for variance reduction in stochastic gradient descent. We denote the collection \( \{f_i\}_{i=1}^n \) of functions that make up \( f \) in (2.1) by \( \mathcal{F} \). For our algorithm, we maintain an additional parameter \( \alpha^t_i \in \mathbb{R}^d \) for each \( f_i \in \mathcal{F} \). We use \( A^t \) to denote \( \{\alpha^t_i\}_{i=1}^n \). The general iterative framework for updating the parameters is presented as Algorithm 3. Observe that the algorithm is still abstract, since it does not specify the subroutine \( \text{SCHEDULEUPDATE} \).

This subroutine determines the crucial update mechanism of \( \{\alpha^t_i\} \) (and thereby of \( A^t \)). As we will see different schedules give rise to different fast first-order methods proposed in the literature. The part of the update based on \( A^t \) is the key for these approaches and is responsible for variance reduction. Figure 3.1 provides different instantiations of the framework and construct a new algorithm derived from it. In particular, we consider incremental methods \( \text{SAG} \) \[56\], \( \text{SVRG} \)
that follow SAGA\textsuperscript{[21]} and SAGA\textsuperscript{[9]}, and classic gradient descent \textsc{GradientDescent} for demonstrating our framework.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| \textsc{SVRG:ScheduleUpdate} \((\{x^i\}_{i=0}^{t+1}, A^t, t, I_T)\) | for \(i = 1 \text{ to } n\) do \[
\alpha^t_{i+1} = 1(m \mid t)x^t + 1(m \mid t)(\alpha^t_i);
\] end return \(A^{t+1}\) |
| \textsc{SAG:ScheduleUpdate} \((\{x^i\}_{i=0}^{t+1}, A^t, t, I_T)\) | for \(i = 1 \text{ to } n\) do \[
\alpha^t_{i+1} = 1(i_{t+1} = i)x^{t+1} + 1(i_{t+1} 
eq i)(\alpha^t_i);
\] end return \(A^{t+1}\) |
| \textsc{SAGA:ScheduleUpdate} \((\{x^i\}_{i=0}^{t+1}, A^t, t, I_T)\) | for \(i = 1 \text{ to } n\) do \[
\alpha^t_{i+1} = 1(i_t = i)x^t + 1(i_t 
eq i)(\alpha^t_i);
\] end return \(A^{t+1}\) |
| \textsc{GD:ScheduleUpdate} \((\{x^i\}_{i=0}^{t+1}, A^t, t, I_T)\) | for \(i = 1 \text{ to } n\) do \[
\alpha^t_{i+1} = x^{t+1};
\] end return \(A^{t+1}\) |

Figure 3.1: \textsc{ScheduleUpdate} function for \textsc{SVRG} (top left), \textsc{SAGA} (top right), \textsc{SAG} (bottom left) and \textsc{GradientDescent} (bottom right). While \textsc{SVRG} is epoch-based, rest of algorithms perform updates at each iteration. Here \(a | b\) denotes that \(a\) divides \(b\).

A straightforward approach to design a new scheduler is to combine the schedules of the above algorithms. This allows us to tradeoff between the various aforementioned parameters of our interest. We call this schedule \textit{hybrid stochastic average gradient} (\textit{HSAG}). Here, we use the schedules of \textsc{SVRG} and \textsc{SAGA} to develop \textit{HSAG}. However, in general, schedules of any of these algorithms can be combined to obtain a hybrid algorithm. Consider some \(S \subseteq [n]\), the indices that follow \textsc{SAGA} schedule. We assume that the rest of the indices follow an \textsc{SVRG}-like schedule with \textit{schedule frequency} \(s_i\) for all \(i \in \bar{S} \triangleq [n] \setminus S\). Now that we have a general framework for \textit{VR} methods, we propose asynchronous variants of the algorithms captured in this framework. There are four key components in our asynchronous algorithm; these are briefly described below.

1. \textbf{Read}: Read the iterate \(x\) and compute the gradient \(\nabla f_{i_t}(x)\) for a randomly chosen \(i_t\).
2. \textbf{Read schedule iterate}: Read the schedule iterate \(A\) and compute the gradients required for update in Algorithm 3.
3. \textbf{Update}: Update the iterate \(x\) with the computed incremental update in Algorithm 3.
4. \textbf{Schedule Update}: Run a scheduler update for updating \(A\).

For the purpose of our analysis, we assume a consistent read model. In particular, our analysis assumes that the vector \(x\) used for evaluation of gradients is a valid iterate that existed at some point in time. Such an assumption typically amounts to using locks in practice. This problem can be avoided by using random coordinate updates as in \textsc{[44]} (see Section 4 of \textsc{[44]}) but such a procedure is computationally wasteful in practice. Let \(\|x\|^2\) denote \(\sum_{j \in c_i} \|x_j\|^2\); then, the convergence depends on \(\Delta\), the smallest constant such that \(\mathbb{E}_i[\|x\|^2] \leq \Delta\|x\|^2\). Intuitively, \(\Delta\) denotes the average frequency with which a feature appears in the data matrix. We are interested in situations where \(\Delta \ll 1\). The following result can be proved for \textit{HSAG} — the key algorithm captured in our general framework.

**Theorem 3.1.1 (Informal)** For appropriate selection of step size \(\eta\) and epoch size \(m\), the iter-
ates of asynchronous variant of Algorithm 3 with HSAG schedule we have
\[ E \left[ f(\tilde{x}^k) - f(x^*) \right] \leq \theta^k_a \left[ f(x^0) - f(x^*) \right], \]
where \( \theta_a \) depends on \( \Delta \) and \( \tau \) (the bound on the delay).

Under appropriate parameter settings and step size normalized by \( \Delta^{1/2} \tau \), we can show linear speedup with \( \tau < 1/\Delta^{1/2} \). Note that \( \tau \) intuitively denotes the parallelism in the asynchronous algorithm. Thus, sparsity is critical for achieving good speedups.

**Randomized Coordinate Descent with Linear Constraints.** We also developed asynchronous variants for randomized coordinate descent with linear constraints. In this setting, we are interested in the following composite objective convex problem with *non-separable* linear constraints

\[
\min_x F(x) := f(x) + h(x) \quad \text{s.t.} \quad Ax = 0.
\]

Here, \( f : \mathbb{R}^d \to \mathbb{R} \) is assumed to be continuously differentiable and convex, while \( h : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\} \) is lower semi-continuous, convex, coordinate-wise separable, but not necessarily smooth; the linear constraints are specified by a matrix \( A \in \mathbb{R}^{m \times d} \), for which \( m \ll d \). In a UAI 2015 paper [45], we develop asynchronous randomized coordinate descent method for solving optimization problems of this form. The reader might wonder the connection between optimization problem (3.3) and finite-sum minimization problems of our interest. However, observe that any finite-sum minimization problem \( \min_x \frac{1}{n} \sum_i f_i(x) \) can be rewritten using variable-splitting as

\[
\min_{\{x_i = x, \forall i \in [N]\}} \frac{1}{n} \sum_{i=1}^{N} f_i(x_i).
\]

Solving the problem in distributed environment requires considerable synchronization (for the consensus constraint), which can slow down the algorithm significantly. However, the dual of the problem is

\[
\min_{\lambda} \frac{1}{n} \sum_{i=1}^{N} f_i^*(\lambda_i) \quad \text{s.t.} \quad \sum_{i=1}^{N} \lambda_i = 0,
\]

where \( f_i^* \) is the Fenchel conjugate of \( f_i \). This reformulation perfectly fits our problem formulation in (3.3) and can be solved in an asynchronous manner using the procedure proposed in our paper. Other interesting application include constrained least square problem, multi-agent planning problems, resource allocation—see [32, 33] and references therein for more examples. We refer interested readers to [45] for more details about the algorithm.

### 3.2 Proposed Work

**Communication Efficient Optimization Algorithms.** In the previous sections of this chapter, we discussed asynchronous optimization algorithms for problems that arise in machine learning. However, the communication efficiency of these algorithms was not examined. In modern
detailed ML systems, machines read and write global parameter frequently. This data access requires massive amount of network bandwidth and hence, communication efficiency of optimization algorithms is critical. To this end, we develop and discuss new communication efficient optimization algorithms.

For the purpose of this discussion, we assume a parameter server architecture for investigating these algorithms. Such a setup entails a server group and a worker group where each group contains several threads/machines. The server machines mainly serve the purpose of maintaining the global parameters, while most of the workload needs to be allocated to the worker machines. The communication between the worker and the server group is the assumed to be the main bottleneck and hence, needs to minimized. We plan to investigate the following two paradigms for address this problem in the context of ML.

1. Iterative algorithms like SGD often require large amount of communication between worker and server machines. The first approach to reduce this communication cost is by constructing a small summary of the training data — which acts as a proxy for the entire data set — and communicating it to the server machine; thereby, eliminating the need for frequent communication between the worker and the server machines. Such an approach entails (i) computing these summaries of data at the worker nodes (which can be computationally intensive), (ii) sending these small summaries to a server machine (a low communication overhead task) and (iii) finally, solving a small optimization problem at the server machines. This summary of the training points is called a coreset. While this methodology has been successfully applied to data clustering problems like k-means and k-median (we refer the reader to [12, 13] for a comprehensive survey), it remains largely unexplored for supervised learning and optimization problems. I have a preliminary algorithm that uses this methodology in the context of solving empirical loss minimization problems for particular loss functions. I plan to investigate this direction further.

2. An alternate approach is to perform most of the computation at the worker machines in an embarrassingly manner and combine the solutions from all the worker machines; thereby, eliminating the need to communicate to the server machine frequently. Note that such an operation needs to be done in an iterative fashion in order to each an optimal solution to our optimization problem. ADMM (alternating direction method of multipliers) is an popular approach for solving distribution optimization problems that follows this methodology. Under certain conditions, ADMM is shown to achieve communication complexity of $O(\sqrt{L/\lambda} \log(1/\epsilon))$ for $L$-smooth and $\lambda$-strongly convex function. More recently, the DANE, DISCO and COCOA+ algorithms have been proposed to tackle the problem of reducing the communication complexity in solving problems of form (2.1) [20, 60, 64]. DISCO is particularly appealing because they match communication complexity lower bounds derived in [4]. However, DISCO requires a second-order oracle for its execution and is not embarrassingly parallel. I am working on developing a first-order algorithm that not only achieves the communication lower bounds in [4] but can be also be implemented in an embarrassingly parallel fashion.
3.2.1 Experiments

For my thesis, I plan to investigate the performance of the proposed algorithms on a few popular ML problems, such as linear and logistic regression, for fairly large datasets. In particular, it will be interesting to understand practical speedups and communication complexities of the proposed algorithms on large sparse datasets.
Chapter 4

My Other Research — In a Nutshell

Besides optimization methods for machine learning applications, I have also worked on a number of other topics such as kernel methods, hypothesis testing, dependence measures, functional regression, which are not part of my thesis. Here, I briefly describe my contributions that form the core of my research in these areas.

**Kernel Methods, Dependence Measures & Hypothesis Testing.** Measuring dependencies and conditional dependencies are of great importance in many scientific fields including machine learning, and statistics. There are numerous problems where we want to know how large the dependence is between random variables, and how this dependence changes if we observe other random variables. In an ICML 2013 paper [46], we developed new kernel based dependence measures with scale invariance property and showed the effectiveness of our dependence measure on real-world problems. I also worked on kernel based two sample testing problem. In particular, in a series of papers [43, 48, 49] we proved (through theoretical and empirical results) that MMD based two sample testing — a kernel based two sample test — also suffers from the curse of dimensionality. Such a result was important because there was a wide misconception that these measures do not suffer from the curse of dimensionality and hence, also work for high dimension problems. Our papers cleared this misconception and provided first theoretical results for the power of the MMD based two sample hypothesis tests.

**Machine Learning on Functional Data.** Another important aspect of many modern machine learning applications is the structure in the input data. Modern data collection techniques motivate settings where the training data is no longer of simple form such as features. For example, a Facebook user profile contains very rich information about the user such as posts, friends list, likes, photos, polls, etc. Unfortunately, most of the existing machine learning and statistical techniques cannot handle such data, often resorting to ad-hoc approaches, thereby ignoring the underlying rich structure in the data. This necessitates the development of a different machine learning paradigm where the true structure in the complex data can be exploited. In a UAI’14 [47], we developed a simple nearest neighbor based algorithm for handling data of various forms. In fact, we considered a strictly generalized scenario of noisy and incomplete/missing data. We analyzed the theoretical properties of our proposed estimator and demonstrated its performance through practical experiments. We also proposed an approach to choose the number of nearest neighbors to be used, thereby alleviating the problem of cross validation in nearest neighbor based algorithms.
Bibliography


Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In NIPS 26, pages 378–385, 2013. 5

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for reg-


