Variations of the Stochastic Gradient Descent for Multi-label Classication Loss Functions

Gaurush Hiranandani gaurush2@illinois.edu

Ping-Ko Chiu pchiu5@illinois.edu

Abstract

Optimization algorithms for large scale learning has gained considerable attention in the recent years in the Machine Learning community. Stochastic Gradient Descent (SGD) methods have become popular in today's world of data abundance. Many variants of SGD has since surfaced to attempt to reduce the variance of the gradients to provide better convergence to the optimal solution. We implement ten different variations (published within the last five years) of the standard stochastic gradient descent and report our findings in this article.

To put the various SGD algorithms in practice, we apply them on a multi-label classification problem with different loss functions. Multi-label classification is one of the most interesting problems in Machine Learning having usefulness in multiple areas / industries, if solved properly. We study two loss functions - Hamming Loss and Subset 0/1 loss along with their similarities and differences. With these two losses, we identify two approaches to the multi-label problem - reduction to independent binary classification problems and reduction to a multi-class classification problem. This report contains related theoretical analysis and results for the two loss functions with ten different variants of SGD implemented on a benchmark multi-label classification dataset.

Contents

1. Introduction

In this project, we work with different variants of SGD, particularly falling in the category of *gradient aggregation* (described later) methods. We investigate and implement ten different very recent variants of the standard SGD method. These are: SVRG by Johnson and Zhang (2013) (three variants), SAGA by Defazio et al. (2014a), SAG by Roux et al. (2012), S2GD by Konen and Richtrik (2017), Finito by Defazio et al. (2014b), VR-Lite by De et al. (2015), Batching SVRG by Harikandeh et al. (2015), and Mixing SVRG by Harikandeh et al. (2015). All of these variants use historical information of the optimization process to provide better gradient estimates to the updates. While they all aim to improve on the basic SGD algorithm, a few of them target specific problems. Some of these algorithms have better convergence rates than others on certain problems. For example, SAGA claims to have a convergence rate improvement of a factor of 2 over SVRG and is directly applicable to non-convex problems without modification. VR-Lite attempts to reduce memory and computation. In addition to implementing these algorithms, we provide the analysis around the lines and implementation findings on these different variations of SGD.

In order to see these variants in action, we dive into Multi-label Classification, which essentially refers to the problem of assigning multiple labels to a given input. Multi-label classification problem shows up in many areas of NLP, image recognition, and signal processing. An important aspect in this problem is to understand the structure behind assignments of multiple labels to an instance. Different kinds of loss functions have been proposed in the literature. For example, Hamming loss, rank loss, subset 0/1 loss, etc. All of them consider the structure in the labels in one way or the another. In this article, we work with two different loss functions - Hamming loss and Subset 0/1 loss. On one hand, there is Hamming loss which can be decomposed over labels and is very much aligned with binary classification. On the other hand, we have subset 0/1 loss which is non-decomposable over the labels, captures the correlation among the labels and very much aligned with multi-class classification in some cases. Since these two loss functions are not differentiable, we identify surrogate loss functions, derive gradient update rules for them, and implement the above mentioned SGD variants.

All the implementation corresponding to SGD variants and loss functions are run on Music by Emotion dataset provided by Trohidis et al. (2011). We chose this data for its small size, making our problems more tractable given our computational power. Since we do no aim to achieve best classification accuracy or the minimum loss, we do not attempt to fine tune the parameters in these algorithms. Rather, we are interested in the process of optimization in these algorithms; therefore, same parameters are used consistently throughout the different variants of SGD to keep the comparison fair.

The structure of the report is as follows. Section 2 contains the problem definition and notations, some results based on the analysis of strongly convex objective functions which are useful for Machine Learning, and discusses why reducing variance

in the gradient is important. In Section 3, we discuss theory related to different loss functions in the multi-label classification setting, observe how different loss functions have different importance, and how they lead to different interpretation of the same problem. We discuss our dataset, implementation methodology, and the results for ten different variants for SGD in Section 4. In Section 5, we provide our conclusions.

2. Background - PART I - Large Scale Optimization

Mathematical optimization in Machine Learning involves, based on currently available data, choosing parameters which are optimal with respect to a given learning problem defined in the sense of minimizing a loss function. The traditional gradient-based methods, which involve a batch approach, have been effective for solving small-scale problems; however, in the large-scale machine learning context, stochastic gradient method (SG) proposed by Robbins and Monro (1951) has been the core strategy of interest. We discuss recent *gradient aggregation* methods, which fall under the *noise reduction* methods, that attempt to borrow some advantages of the batch methods, such as their fast convergence rates. [1]

2.1 Problem Definition and Notations

We define our problem in terms of multi-label classification setting. Our goal is to determine a prediction function $h : \mathcal{X} \to \mathcal{Y} \in \{0,1\}^M$ on an input space $\mathcal{X}$ to an output space $\mathcal{Y}$ such that, given $x \in \mathcal{X}$, the value $h(x)$ offers an accurate prediction about the true output y. To do this, one should choose the prediction function h by attempting to minimize a risk measure over an adequately selected family of prediction functions, call it $\mathcal{H}$. We sometimes abuse notation and treat h to be a vector valued prediction function, depending on the context. Similarly, we denote x and y as vectors $\mathbf{x}$ and $\mathbf{y}$, respectively, depending on the context.

We assume that the prediction function h has a fixed form and is parameterized by a real vector $w \in \mathbb{R}^d$ over which the optimization is to be performed. In this project, we take the family of linear functions, that is, our prediction function takes the form $w^T x$. In general, for some given $h(\cdot;\cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^d \to \mathbb{R}^{d_y}$, we consider the family of prediction functions

$$\mathcal{H} := \{h(\cdot; w) : w \in \mathbb{R}^d\}$$

We aim to find the prediction function in this family which minimizes certain losses incurred from inaccurate predictions. We assume a loss function denoted by $\ell : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \to \mathbb{R}$ as one that, given an input-output pair (x, y), yields the loss $\ell(h(x; w); y)$ when $h(x; w)$ and y are the predicted and true outputs, respectively.

The parameter vector w is chosen to minimize the expected loss that would be incurred from any input-output pair. We assume that losses are measured with respect

1. The theoretical analysis is borrowed from Bottou et al. (2016), but written succinctly depending on the scope of this project. The proofs can be found in the same source.

to a probability distribution $P(x, y)$. The objective function we wish to minimize is

$$R(w) = \int_{\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}} \ell(h(x; w); y) dP(x; y) = \mathbb{E}[\ell(h(x; w); y)] \tag{2.1}$$

We say that $R : \mathbb{R} \to \mathbb{R}$ yields the risk (i.e., expected loss) given a parameter vector w with respect to the probability distribution P.

In reality, P is usually unknown. Thus, in practice, we work with an estimate of the expected risk R. We take the set of $n \in \mathbb{N}$ independently drawn input-output samples $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, with which one may define the empirical risk function $R_n : \mathbb{R}^d \to \mathbb{R}$ by

$$R_n(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w); y_i) \tag{2.2}$$

Let us represent a sample (or set of samples) by a random seed ξ. In this entire document, we refer to the loss incurred for a given $(w; \xi)$ as $f(w; \xi)$, i.e.,

f is the composition of the loss function ℓ and the prediction function h.

From the above definition of loss composed with prediction function, we have the expected risk for a given w as:

$$R(w) = E[f(w; \xi)], \tag{2.3}$$

loss incurred by the parameter vector w with respect to the i^{th} sample as:

$$f_i(w) := f(w; \xi_{[i]}), \tag{2.4}$$

and a sample estimator of expected risk as:

$$R_n(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w). \tag{2.5}$$

We use [i] to denote the i^{th} element of a fixed set of realizations of a random variable, and k to denote the k^{th} element of a sequence of random variables.

2.2 Stochastic vs. Batch Gradient Descent

Optimization methods for machine learning fall into two broad categories - *stochastic* and *batch*. The stochastic optimization method is the stochastic gradient method (SG), where the $(k + 1)^{th}$ iterate is defined as:

$$w_{k+1} \leftarrow w_k - \alpha_k \nabla f_{i_k}(w_k) \tag{2.6}$$

Here, for all $k \in \mathbb{N}$, the index i_k (corresponding to the seed $\xi_{[i_k]}$, i.e., the sample pair $(x_{i_k}; y_{i_k})$ is chosen randomly from $\{1, ..., n\}$ and α_k is a positive stepsize. While each direction $\nabla f_{i_k}(w_k)$ might not be one of descent from w_k, if it is a descent direction in expectation, then the sequence $\{w_k\}$ can be guided toward a minimizer of R_n.

The other commonly used approach is the batch approach, aka, full gradient (FG) method, where the parameters at the $(k+1)^{th}$ iteration is defined as:

$$w_{k+1} \leftarrow w_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} \nabla f_{i_k}(w_k) \tag{2.7}$$

Computing the step $\sum_{i=1}^{n} \nabla f_{i_k}(w_k)$ in FG is more expensive than computing the step $\nabla f_{i_k}(w_k)$ in SG, though one may expect that a better step is computed when all samples are considered in an iteration.

In this report, we investigate the *gradient aggregation* methods which aim to improve the rate of convergence from sublinear to linear. These methods do not compute mini-batches of fixed size, nor do they compute full gradients in every iteration. Instead, they dynamically incorporate new gradient information in order to construct a more reliable step with smaller variance than an SG step. These are considered to be one of the ways to reduce *noise* in the gradient direction. Hence, they fall under the category of *noise reduction* methods.

2.3 Why Reducing Variance is Important!

We first discuss the theoretical results that we can get without making strong convexity assumption on the objective function. Then we deep dive on the results which are attained under the setting of strongly convex objective. The analysis as well as implementation in 4 is based on minimizing a strongly convex objective function. This is because, it is possible to establish a global rate of convergence to the optimal objective value in these cases.

2.3.1 Preliminary SG Analysis

The basic SGD algorithm looks as given in Algorithm 2.1.

Algorithm 2.1 Stochastic Gradient Descent (SGD) Algorithm

1: Choose an initial iterate w_1.
2: **for** $k = 1, 2, ...$ do **do**
3: Generate a realization of the random variable ξ_k.
4: Compute a stochastic vector $g(w_k, \xi_k)$.
5: Choose a stepsize $\alpha_k > 0$.
6: Set the new iterate as $w_{k+1} \leftarrow w_k - \alpha_k g(w_k, \xi_k)$.
7: **end for**

The Algorithm 2.1 is in a very general sense. First, the value of the random variable ξ_k need only be viewed as a seed for generating a stochastic direction; as

such, a realization of it may represent the choice of a single training sample as in the simple SG method, or may represent a set of samples as in the mini-batch SG method. Second, $g(w_k, \xi_k)$ could represent a stochastic gradient - i.e., an unbiased estimator of $\nabla F(w_k)$ most of the times, except in very few algorithms that are studied in literature. All the theoretical results hold as long as the expected angle between $g(w_k, \xi_k)$ and $\nabla F(w_k)$ is sufficiently positive. Third, Algorithm 2.1 allows various choices of the stepsize sequence $\{\alpha_k\}$; however, we focus on only fixed step sizes. We refer to Algorithm 2.1 as SG. The particular instance 2.6 is referred to as basic SG, whereas the instance 2.7 will be referred to as mini-batch SG.

The theory related to the analysis of stochastic gradient descent require us to make some assumptions. The first assumption is as follows:

Assumption 2.1 (Lipschitz-continuous objective gradients) *The objective function $F : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable and the gradient function of F, namely, $\nabla F : \mathbb{R}^d \to \mathbb{R}^d$, is Lipschitz continuous with Lipschitz constant $L > 0$, i.e.,*

$$\|\nabla F(w) - \nabla F(\overline{w})\|_2 \leq L\|w - \overline{w}\|_2 \quad \textit{for all} \quad \{w, \overline{w}\} \subset \mathbb{R}^d.$$

We call such functions as L-smooth functions. This assumption tells us that the gradient of F does not change arbitrarily quickly with respect to the parameter vector. Following directly from Assumption 2.1, we have

$$F(w) \leq F(\overline{w}) + \nabla F(\overline{w})^T(w - \overline{w}) + \tfrac{1}{2}L\|w - \overline{w}\|_2^2 \quad \text{for all} \quad \{w, \overline{w}\} \subset \mathbb{R}^d. \tag{2.8}$$

Using the above result 2.8, we have the following Lemma.

Lemma 2.2 *Under Assumption 2.1, the iterates of SG (Algorithm 2.1) satisfy the following inequality for all $k \in \mathbb{N}$:*

$$\mathbb{E}_{\xi_k}[F(w_{k+1})] - F(w_k) \leq -\alpha_k \nabla F(w_k)^T \mathbb{E}_{\xi_k}[g(w_k, \xi_k)] + \tfrac{1}{2}\alpha_k^2 L \mathbb{E}_{\xi_k}[\|g(w_k, \xi_k)\|_2^2]. \tag{2.9}$$

This lemma shows that, regardless of how SG arrived at w_k, the expected decrease in the objective function yielded by the kth step is bounded above by a quantity involving: (i) the expected directional derivative of F at w_k along $-g(x_k, \xi_k)$ and (ii) the second moment of $g(x_k, \xi_k)$. For example, if $g(w_k, \xi_k)$ is an unbiased estimate of $\nabla F(w_k)$, then it follows from Lemma 2.2 that

$$\mathbb{E}_{\xi_k}[F(w_{k+1})] - F(w_k) \leq -\alpha_k \|\nabla F(w_k)\|_2^2 + \tfrac{1}{2}\alpha_k^2 L \mathbb{E}_{\xi_k}[\|g(w_k, \xi_k)\|_2^2]. \tag{2.10}$$

In the following analysis, it is shown that convergence of SG is guaranteed as long as the stochastic directions and stepsizes are chosen such that the right-hand side of (2.9) is bounded above by a deterministic quantity that asymptotically ensures sufficient descent in F. This is ensured by adding more constraints on the first and second moments of the stochastic directions $\{g(w_k, \xi_k)\}$. Therefore, in order to limit the harmful effect of the right most term in (2.10), we restrict the variance of $g(w_k, \xi_k)$, i.e.,

$$\mathbb{V}_{\xi_k}[g(w_k, \xi_k)] := \mathbb{E}_{\xi_k}[\|g(w_k, \xi_k)\|_2^2] - \|\mathbb{E}_{\xi_k}[g(w_k, \xi_k)]\|_2^2 . \tag{2.11}$$

Assumption 2.3 *We assume that the objective function and SG (Algorithm 2.1) satisfy the following:*

(a) The sequence of iterates $\{w_k\}$ is contained in an open set over which F is bounded below by a scalar F_{inf}.

(b) There exist scalars $\mu_G \geq \mu > 0$ such that, for all $k \in \mathbb{N}$,

$$\nabla F(w_k)^T \mathbb{E}_{\xi_k}[g(w_k, \xi_k)] \geq \mu \|\nabla F(w_k)\|_2^2 \quad \text{and} \tag{2.12a}$$

$$\|\mathbb{E}_{\xi_k}[g(w_k, \xi_k)]\|_2 \leq \mu_G \|\nabla F(w_k)\|_2. \tag{2.12b}$$

(c) There exist scalars $M \geq 0$ and $M_V \geq 0$ such that, for all $k \in \mathbb{N}$,

$$\mathbb{V}_{\xi_k}[g(w_k, \xi_k)] \leq M + M_V \|\nabla F(w_k)\|_2^2. \tag{2.13}$$

The first assumption 2.3(a) requires that the objective function to be bounded below over the domain explored by the algorithm. The second assumption 2.3(b), states that, in expectation, the vector $-g(w_k, \xi_k)$ is in a direction of sufficient descent for F from w_k with a norm comparable to the norm of the gradient. The third assumption 2.3(c), states that the variance of $g(w_k, \xi_k)$ is restricted.

All together, Assumption 2.3, combined with the definition (2.11), requires that the second moment of $g(w_k, \xi_k)$ satisfies

$$\mathbb{E}_{\xi_k}[\|g(w_k, \xi_k)\|_2^2] \leq M + M_G \|\nabla F(w_k)\|_2^2 \quad \text{with} \quad M_G := M_V + \mu_G^2 \geq \mu^2 > 0. \tag{2.14}$$

Using the above assumptions 2.3 further, we get the following lemma which builds on Lemma 2.2.

Lemma 2.4 *Under Assumptions 2.1 and 2.3, the iterates of SG (Algorithm 2.1) satisfy the following inequalities for all $k \in \mathbb{N}$:*

$$\mathbb{E}_{\xi_k}[F(w_{k+1})] - F(w_k) \leq -\mu\alpha_k \|\nabla F(w_k)\|_2^2 + \tfrac{1}{2}\alpha_k^2 L \mathbb{E}_{\xi_k}[\|g(w_k, \xi_k)\|_2^2] \tag{2.15a}$$

$$\leq -(\mu - \tfrac{1}{2}\alpha_k L M_G)\alpha_k \|\nabla F(w_k)\|_2^2 + \tfrac{1}{2}\alpha_k^2 L M. \tag{2.15b}$$

This lemma reveals that regardless of how the method arrived at the iterate w_k, the optimization process continues in a Markovian manner. The parameter iterate w_{k+1} is a random variable that depends only on the iterate w_k, the seed ξ_k, and the stepsize α_k. It does not depend on the past iterates. This is indeed true as the difference $\mathbb{E}_{\xi_k}[F(w_{k+1})] - F(w_k)$ is bounded above by a deterministic quantity. Furthermore, the first term in (2.15b) is strictly negative for small α_k and suggests a decrease in the objective function by a magnitude proportional to $\|\nabla F(w_k)\|_2^2$. However, the second term in (2.15b) could be large enough to allow the objective value to increase and hence, balancing these terms is critical in the design of SG methods.

2.3.2 SG Analysis for Strongly Convex Objective

Assumption 2.5 (Strong convexity) *The objective function $F : \mathbb{R}^d \to \mathbb{R}$ is strongly convex in that there exists a constant $c > 0$ such that*

$$F(\overline{w}) \geq F(w) + \nabla F(w)^T(\overline{w} - w) + \tfrac{1}{2}c\|\overline{w} - w\|_2^2 \quad \text{for all} \quad (\overline{w}, w) \in \mathbb{R}^d \times \mathbb{R}^d. \tag{2.16}$$

Hence, F has a unique minimizer, denoted as $w_* \in \mathbb{R}^d$ with $F_* := F(w_*)$.

A useful fact from convex analysis is that if the objective function is strongly convex, then one can bound the optimality gap at a given point in terms of the squared ℓ_2-norm of the gradient of the objective at that point. Formally, we can write:

$$2c(F(w) - F_*) \leq \|\nabla F(w)\|_2^2 \quad \text{for all} \quad w \in \mathbb{R}^d. \tag{2.17}$$

Furthermore, from (2.8) and (2.16), the constants in Assumptions 2.1 and 2.5 must satisfy $c \leq L$. This is important in order to get various convergence rates.

Now, we state the convergence theorem for SG in the case of strongly convex objective function with a fixed stepsize. An interesting thing to note here is that, it does not prove convergence to the solution, but only to a neighborhood of the optimal solution.

Theorem 2.6 (Strongly Convex Objective, Fixed Stepsize) *Under Assumptions 2.1, 2.3, and 2.5 (with $F_{\text{inf}} = F_*$), suppose that the SG method is run with a fixed stepsize, $\alpha_k = \bar{\alpha}$ for all $k \in \mathbb{N}$, satisfying*

$$0 < \bar{\alpha} \leq \frac{\mu}{LM_G}. \tag{2.18}$$

Then, the expected optimality gap satisfies the following inequality for all $k \in \mathbb{N}$:

$$\begin{aligned} \mathbb{E}[F(w_k) - F_*] &\leq \frac{\bar{\alpha}LM}{2c\mu} + (1 - \bar{\alpha}c\mu)^{k-1}\left(F(w_1) - F_* - \frac{\bar{\alpha}LM}{2c\mu}\right) \\ &\xrightarrow{k\to\infty} \frac{\bar{\alpha}LM}{2c\mu}. \end{aligned} \tag{2.19}$$

If $g(w_k, \xi_k)$ is an unbiased estimate of $\nabla F(w_k)$, then $\mu = 1$, and if there is no noise in $g(w_k, \xi_k)$, then we can take $M_G = 1$ (due to (2.14)). In this case, (2.18) reduces to $\bar{\alpha} \in (0, 1/L]$. This is a classical stepsize requirement for a steepest descent method.

Theorem 2.6 reflects on the relationship between the stepsizes and bound on the variance of the stochastic directions. If there were no noise in the gradient computation , then one can obtain linear convergence to the optimal value. We still can use a fixed stepsize and be sure that the expected objective values will converge linearly to a neighborhood of the optimal value. This is indeed quite useful in many practical applications. Furthemore, after some point, the noise in the gradient estimates prevent further progress. From (2.19), we can see that selecting a smaller stepsize

worsens the contraction constant in the convergence rate, but allows us to reach a solution closer to the optimal value.

Critical Remark: An algorithm that optimizes the empirical risk R_n has access to an additional piece of information: it knows when a gradient estimate is evaluated on a training example that has already been visited during previous iterations. Recent *gradient aggregation* methods (see §4) make use of this information and improve upon the lower bound for the optimization of the empirical risk (though not for the expected risk). These algorithms enjoy linear convergence with low computing times in practice and the primary motivation behind this project.

2.4 Noise Reduction Analysis for SGD

As discussed above, SG suffers from the adverse effect of noisy gradient estimates. This prevents it from converging to the solution or achieve a slow, sublinear rate of convergence when fixed stepsizes are used. Many methods, proven to be effective in practice and enjoying attractive theoretical properties, have been developed to address this limitation. These methods usually reduce the errors in the gradient estimates and/or iterate sequence. In this report, we discuss ten different variations of SGD which fall under the category of *gradient aggregation* method. *Gradient aggregation* methods improve the quality of the search directions by storing gradient estimates corresponding to samples employed in previous iterations, updating one (or some) of these estimates in each iteration, and defining the search direction as a weighted average of these estimates.

We first discuss the fundamental result that stipulates a rate of decrease in noise that allows a stochastic-gradient-type method to converge at a linear rate. Next, we discuss the ten gradient aggregation methods in Section 4.

Let us recall the fundamental inequality (2.9):

$$\mathbb{E}_{\xi_k}[F(w_{k+1})] - F(w_k) \leq -\alpha_k \nabla F(w_k)^T \mathbb{E}_{\xi_k}[g(w_k, \xi_k)] + \tfrac{1}{2}\alpha_k^2 L \mathbb{E}_{\xi_k}[\|g(w_k, \xi_k)\|_2^2].$$

We can see that if $-g(w_k, \xi_k)$ is a descent direction in expectation and if we are able to decrease $\mathbb{E}_{\xi_k}[\|g(w_k, \xi_k)\|_2^2$ fast enough, then the effect of having noisy directions will not impede a fast rate of convergence. From different perspective, we can expect the described behavior if, in Assumption 2.3, the variance of $g(w_k, \xi_k)$ vanishes sufficiently quickly.

Theorem 2.7 (Strongly Convex Objective, Noise Reduction) *Suppose that Assumptions 2.1, 2.3, and 2.5 (with $F_{\text{inf}} = F_*$) hold, but with (2.13) refined to the existence of constants $M \geq 0$ and $\zeta \in (0, 1)$ such that, for all $k \in \mathbb{N}$,*

$$\mathbb{V}_{\xi_k}[g(w_k, \xi_k)] \leq M\zeta^{k-1}. \tag{2.20}$$

*In addition, suppose that the SG method (Algorithm **??**) is run with a fixed stepsize, $\alpha_k = \bar{\alpha}$ for all $k \in \mathbb{N}$, satisfying*

$$0 < \bar{\alpha} \leq \min\left\{\frac{\mu}{L\mu_G^2}, \frac{1}{c\mu}\right\}. \tag{2.21}$$

Then, for all $k \in \mathbb{N}$, the expected optimality gap satisfies

$$\mathbb{E}[F(w_k) - F_*] \leq \omega \rho^{k-1}, \tag{2.22}$$

where

$$\omega := \max\{\tfrac{\bar{\alpha} LM}{c\mu}, F(w_1) - F_*\} \tag{2.23a}$$

$$\textit{and} \quad \rho := \max\{1 - \tfrac{\bar{\alpha} c \mu}{2}, \zeta\} < 1. \tag{2.23b}$$

Consideration of the typical magnitudes of the constants μ, L, μ_G, and c in (2.21) reveals that the admissible range of values of $\bar{\alpha}$ is large, i.e., the restriction on the stepsize $\bar{\alpha}$ is not unrealistic in practical situations.

In order to see SGD in action, we need to define loss functions. Hence, we discuss important theory related to multi-label classification losses.

3. Background - PART II - Multi-label Classification

We are also interested in exploring loss functions for multi-label classification problem. Hence, digressing a little from the above discussion of Stochastic Gradient Descent, we discuss some of the properties of the loss functions in the multi-label classification (MLC) setting. We use these loss functions with the variants of stochastic gradient descent in Section 4.

A trivial approach to solve MLC problems can be through decomposition into several binary classification problems; one binary classifier can be trained for each label and used to predict whether this label is present (relevant) or not in the instance. However, this approach has been criticized for ignoring information about the interdependencies between the labels which can be crucial in some cases. Since we aim to predict all the labels simultaneously, it is important to exploit any such dependencies.

We analyze two specific but representative loss functions, namely the Hamming loss and the subset 0/1 loss. Hamming loss does not consider the dependence amongst the labels, whereas the subset 0/1 loss does [2].

We use the same setup as described in 2.1. We will denote some of the additional notations as follows. Let $\mathcal{L} = \lambda_1, \lambda_2, ..., \lambda_m$ be a finite set of class labels. We assume that an instance $\mathbf{x} \in \mathcal{X}$ is (non-deterministically) associated with a subset of labels $L \in 2^{\mathcal{L}}$; this subset is often called the set of relevant labels, while the complement $L \setminus \mathcal{L}$ is considered as irrelevant for $\mathbf{x}$. We identify a set L of relevant labels with a binary vector $\mathbf{y} = (y_1, y_2, ..., y_m)$, in which $y_i = 1 \iff \lambda_i \in L$. By $\mathcal{Y} = \{0, 1\}^m$ we denote the set of possible labellings. We assume observations to be generated independently and identically according to a probability distribution $P(\mathbf{X}, \mathbf{Y})$ on $\mathcal{X} \times \mathcal{Y}$, i.e., an observation $\mathbf{y} = (y_1, ..., y_m)$ is a realization of a corresponding random

2. The theoretical analysis is borrowed from Dembczyński et al. (2012), but written succinctly depending on the scope of this project. The proofs can be found in the same source.

vector $\mathbf{Y} = (Y_1, Y_2, ..., Y_m)$. We denote by $P(\mathbf{y}|\mathbf{x})$ the conditional distribution of $\mathbf{Y} = \mathbf{y}$ given $\mathbf{X} = \mathbf{x}$, and by $P(Y_i = b|\mathbf{x})$ the corresponding marginal distribution of Y_i :

$$P(Y_i = b|\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}: y_i = b} P(\mathbf{y}|\mathbf{x}).$$

In general, a multi-label classifier h is an $\mathcal{X} \to \mathbb{R}^m$ mapping that for a given instance $x \in \mathcal{X}$ returns a vector

$$\mathbf{h}(\mathbf{x}) = (h_1(x), h_2(x), ..., h_m(x)).$$

The problem of MLC can then be stated as follows: Given training data in the form of a finite set of observations $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, drawn independently from $P(\mathbf{X}, \mathbf{Y})$, the goal is to learn a classifier $\mathbf{h} : \mathcal{X} \to \mathbb{R}^m$ that generalizes well beyond these observations in the sense of minimizing the risk with respect to a specific loss function. The risk of a classifier $\mathbf{h}$ is defined formally as the expected loss over the joint distribution $P(\mathbf{X}, \mathbf{Y})$:

$$R_L(\mathbf{h}) = \mathbb{E}_{\mathbf{XY}} L(\mathbf{Y}, \mathbf{h}(\mathbf{X})) \tag{3.24}$$

where $L(\cdot)$ is a loss function on multi-label predictions. The so-called risk-minimizing model $\mathbf{h}^*$ is given by

$$\mathbf{h}^* = \arg\min_{\mathbf{h}} \mathbb{E}_{\mathbf{XY}} L(\mathbf{Y}, \mathbf{h}(\mathbf{X})) = \arg\min_{\mathbf{h}} \mathbb{E}_{\mathbf{X}} \mathbb{E}_{\mathbf{Y}|\mathbf{X}} L(\mathbf{Y}, \mathbf{h}(\mathbf{X})) \tag{3.25}$$

and determined in a pointwise way by the risk minimizer

$$\mathbf{h}^*(\mathbf{x}) = \arg\min_{\mathbf{y}} \mathbb{E}_{\mathbf{Y}|\mathbf{X}} L(\mathbf{Y}, \mathbf{y}) \tag{3.26}$$

Usually, the image of a classifier h is restricted to $\mathcal{Y}$, which means that it assigns a predicted label subset to each instance $x \in \mathcal{X}$. However, for some loss functions like subset 0/1 loss, which corresponds to slightly different tasks like ranking or probability estimation, the prediction of a classifier is not limited to binary vectors as we see in Section 3.3.

3.1 Two views on multi-label classification

In this section, we observe a link between label dependence and loss minimization. As shown in the theoretical analysis of Section 3.2, this link is quite natural, since the discussion of the dependence of error terms boils down to a discussion about loss functions. In terms of loss minimization, there are two views prevalent in the literature. These are:

1. The individual label view: How to improve the predictive accuracy of a single label by using information about other labels?

2. The joint label view: What type of proper (non-decomposable) MLC loss functions is suitable for evaluating a multi-label prediction as a whole, and how to minimize such loss functions?

More generally, the questions relate to problems in which, on one hand, the goal is to minimize a loss function that is label-wise decomposable, and on the other hand, the goal is to minimize a non-decomposable loss function.

The simplest loss function in the earlier setting is the Hamming loss, which is defined as the fraction of labels whose relevance is incorrectly predicted:

$$L_H(\mathbf{y}, \mathbf{h}(\mathbf{x})) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}[y_i \neq h_i(\mathbf{x})] \tag{3.27}$$

For the Hamming loss 3.27, the risk minimizer 3.26 is obtained by

$$\mathbf{h}_H^*(\mathbf{x}) = (h_{H_1}(\mathbf{x}), ..., h_{H_m}(\mathbf{x})),$$

where

$$h_{H_i}(\mathbf{x}) = \arg\max_{b \in \{0,1\}} P(Y_i = b|\mathbf{x}) \quad (i = 1, ..., m). \tag{3.28}$$

From this analysis, we can conclude that it is enough to take the marginal (single label) distribution $P(Y_i|\mathbf{x})$ into account in order to solve the problem.

In the second setting, we discuss subset 0/1 loss. The subset 0/1 loss, which is closely related to the estimation of the joint probability distribution, is defined as follows:

$$L_s(y, h(x)) = \mathbb{1}[y \neq h(x)] \tag{3.29}$$

This loss function is quite stringent, especially in the case of many labels. Further, it does not discriminate well between almost correct and completely wrong predictions.

As shown in Dembczyński et al. (2012), the risk-minimizing prediction for 3.29 is simply given by the mode of the distribution:

$$\mathbf{h}_s^*(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y}|\mathbf{x}) \tag{3.30}$$

This shows that the entire distribution of $\mathbf{Y}$ given $\mathbf{X}$, or at least enough knowledge to identify the mode of this distribution, is needed to minimize the subset 0/1 loss. The derivation of a risk-minimizing prediction requires the modeling of the joint distribution (at least to some extent), and hence the modeling of conditional dependence between labels.

3.2 Theoretical insights into multi-label classification

A classifier supposed to be good for solving one of those problems may perform poorly for another problem. We discuss two losses - Hamming loss and subset 0/1 loss. The first one is representative of the single label scenario, while the second one is a typical multi-label loss function whose minimization calls for an estimation of the joint distribution. For the analysis, we take unconstrained hypothesis space, which allows us to consider the conditional distribution for a given $\mathbf{x}$.

In this section, we show that, in general, the Hamming loss minimizer and the subset 0/1 loss minimizer will differ significantly. That is, the Hamming loss minimizer

may be poor in terms of the subset 0/1 loss and vice versa. Therefore, it becomes important to choose our algorithm in compliance with the loss functions.

Theorem 3.8 *The Hamming loss and subset 0/1 have the same risk minimizer, i.e.,* $\mathbf{h}^*_H(\mathbf{x}) = \mathbf{h}^*_s(\mathbf{x})$*, if one of the following conditions holds:*

1. *Labels* $Y_1, ..., Y_m$ *are conditionally independent, i.e.,* $P(\mathbf{Y}|\mathbf{x}) = \prod_{i=1}^m P(Y_i|x)$.
2. *The probability of the mode of the joint probability is greater than or equal to 0.5, i.e.,* $P(\mathbf{h}^*_s(\mathbf{x})|\mathbf{x}) \geq 0.5$.

Furthermore, the two loss functions are related to each other because of the following bounds.

Theorem 3.9 *For all distributions of* $\mathbf{Y}$ *given* $\mathbf{x}$*, and for all models* $\mathbf{h}$*, the expectation of the subset 0/1 loss can be bounded in terms of the expectation of the Hamming loss as follows:*

$$\frac{1}{m}\mathbb{E}_{\mathbf{Y}}[L_s(\mathbf{Y}, \mathbf{h}(\mathbf{x}))] \leq E_{\mathbf{Y}}[L_H(\mathbf{Y}, \mathbf{h}(\mathbf{x}))] \leq E_{\mathbf{Y}}[L_s(\mathbf{Y}, \mathbf{h}(\mathbf{x}))]. \tag{3.31}$$

However, the next set of results show that using a classifier tailored for the wrong loss function may yield bad performance for the other loss. We define the regret of a classifier $\mathbf{h}$ with respect to a loss function L_z as follows:

$$r_{L_z}(\mathbf{h}) = R_{L_z}(\mathbf{h}) - R_{L_z}(\mathbf{h}^*_z) \tag{3.32}$$

where R is the risk given by 3.24, and $\mathbf{h}^*_z$ is the Bayes-optimal classifier with respect to the loss function L_z. The regret with respect to the Hamming loss, given by

$$r_H(\mathbf{h}) = \mathbb{E}_{\mathbf{XY}}[L_H(\mathbf{Y}, \mathbf{h}(\mathbf{X}))] - \mathbb{E}_{\mathbf{XY}}[L_H(\mathbf{Y}, \mathbf{h}^*_H(\mathbf{X}))], \tag{3.33}$$

and the subset 0/1 loss, given by

$$r_s(\mathbf{h}) = \mathbb{E}_{\mathbf{XY}}[L_s(\mathbf{Y}, \mathbf{h}(\mathbf{X}))] - \mathbb{E}_{\mathbf{XY}}[L_s(\mathbf{Y}, \mathbf{h}^*_s(\mathbf{X}))], \tag{3.34}$$

Since both the loss functions are decomposable with respect to individual instances, we analyze the expectation over $\mathbf{Y}$ for a given $\mathbf{x}$. The first result concerns the highest value of the regret in terms of the subset 0/1 loss for $\mathbf{h}^*_H(\mathbf{X})$, the optimal strategy for the Hamming loss.

Theorem 3.10 *The following upper bound holds:*

$$\mathbb{E}_{\mathbf{Y}}[L_s(\mathbf{Y}, \mathbf{h}^*_H(\mathbf{x})] - \mathbb{E}_{\mathbf{Y}}[L_s(\mathbf{Y}, \mathbf{h}^*_s(\mathbf{x})] < 0.5. \tag{3.35}$$

Moreover, this bound is tight, i.e.,

$$\sup_P \mathbb{E}_{\mathbf{Y}}[L_s(\mathbf{Y}, \mathbf{h}^*_H(\mathbf{x})] - \mathbb{E}_{\mathbf{Y}}[L_s(\mathbf{Y}, \mathbf{h}^*_s(\mathbf{x})] < 0.5, \tag{3.36}$$

where the supremum is taken over all probability distributions on $\mathcal{Y}$*.*

The second result concerns the highest value of the regret in terms of the Hamming loss for $\mathbf{h}_s^*(\mathbf{X})$, the optimal strategy for the subset 0/1 loss.

Theorem 3.11 *The following upper bound holds for $m > 3$:*

$$\mathbb{E}_{\mathbf{Y}}[L_H(\mathbf{Y}, \mathbf{h}_s^*(\mathbf{x})] - \mathbb{E}_{\mathbf{Y}}[L_H(\mathbf{Y}, \mathbf{h}_H^*(\mathbf{x})] < \frac{m-2}{m+3}. \tag{3.37}$$

Moreover, this bound is tight, i.e.,

$$\sup_P \mathbb{E}_{\mathbf{Y}}[L_H(\mathbf{Y}, \mathbf{h}_s^*(\mathbf{x})] - \mathbb{E}_{\mathbf{Y}}[L_H(\mathbf{Y}, \mathbf{h}_H^*(\mathbf{x})] = \frac{m-2}{m+3}, \tag{3.38}$$

where the supremum is taken over all probability distributions on $\mathcal{Y}$.

As we can see, the worst case regret is high for both loss functions, suggesting that a single classifier will not be able to perform equally well in terms of both the loss functions. A classifier specifically tailored for the Hamming (subset 0/1) loss will indeed perform much better for this loss than a classifier trained to minimize the subset 0/1 (Hamming) loss.

3.3 Approach for Subset 0/1 Loss

Label Powerset (LP): This approach reduces the MLC problem to multi-class classification, considering each label subset $L \in \mathcal{L}$ as a distinct meta-class. The number of these meta-classes can be $|L| = 2^m$, although it is often reduced considerably by ignoring label combinations that never occur in the training data and hence, the actual number of classes are pretty less. This can be seen in Section 4. LP is tailored for the subset 0/1 loss, since prediction of the most probable meta-class is equivalent to prediction of the mode of the joint label distribution.

4. Implementation and Results - Variants and Loss Functions

We first describe the setup for the conducted experiments and then we look at each of the variants separately.

4.1 Dataset

We have used the music-emotions dataset Trohidis et al. (2008), available from KEEL.[3]. This is a well-known dataset for the automated detection of emotion in music tailored as a multi-label classification task. Here, a piece of music may belong to more than one class. For the feature extraction process, the Marsyas tool Tzanetakis and Cook (2002) was used. The extracted features fall into two categories: rhythmic and timbre. The songs are labeled with emotions by three human judges using the Tellegen-Watson-Clark Yang and Lee (2004) model. There are 593 instances of songs with 72 real valued features. Each instance is labeled with multiple labels from a set of 6 labels.

3. http://sci2s.ugr.es/keel/dataset_smja.php?cod=922#sub1

4.2 Loss Functions and Their Surrogates

We work with linear prediction functions in this report i.e. $\mathcal{H}$ comprises of linear functions of the features $x \in \mathcal{X}$. Therefore, $h(x) = w^T x$. Below, we describe the surrogate loss functions used in this project. Please note that we define the function to be loss composed with the prediction function. That is, $f = \ell \circ h$.

1. Hamming Loss: The hamming loss is defined in 3.27. However, this loss is non-convex and non-differentiable. Therefore, we pick a surrogate loss which is decomposable over labels, similar to Hamming loss. We work with regularized Logistic loss for multi-label classification, which is defines as follows:

$$f(\mathbf{w}; x, y) = \frac{1}{M} \sum_{i=1}^{M} \Big\{ \ln(1 + \exp(-y w_i^T x)) - \lambda_i ||w_i||^2 \Big\} \tag{4.39}$$

   Here, we assume that $y \in \{-1, 1\}$ and w_i is the weight vector corresponding to label i. This make the objective function to be strongly convex and the analysis presented in Section 2.3.2 goes through, which is also evident from the results discussed in Section 4.

2. Subset 0/1 Loss: The subset 0/1 loss is defined in 3.29. Again, this loss is non-convex and non-differentiable. Therefore, we pick a surrogate loss, but this time, it is non-decomposable over labels. Before that, we convert the MLC problem in multi-class classification problem by the Label Powerset (Section 3.3) method. On the given dataset, we got 27 unique combinations of labels. Hence, our problem reduced to multi-class classification with 27 classes. We used the cross entropy loss for the LP setting, which is defined as follows:

$$f(\mathbf{w}; x, y) = -\sum_{i=1}^{K} y_k \log \Big(\frac{e^{w_k^T x}}{\sum_{j=1}^{K} e^{w_i^T x}} \Big) \tag{4.40}$$

   Here, $y_k \in \{0, 1\}$ for $k \in \{1, 2, ..., K\}$ and w_k^T represents the row of the weight matrix $\mathbf{w}$ for label $k \in \{1, 2, ..., K\}$. Here, $K = 27$ for the actual data.

4.3 Variants of Gradient Aggregation Methods and Results on Hamming Loss

We will first investigate the variants of these SGD methods on the hamming loss for multilabel classification problem. For the k labels of the multilabel classification problem, we define k individual binary classification problems to be solved with l_2 regularized Logistic Regression as describes in 4.2.

For all the experiments that will follow, we will adopt a constant stepsize and use "effective passes" to fix a reference frame for the different algorithms. One of the advantages of these gradient aggregation algorithms is that linear convergence is

guaranteed with constant stepsizes when SGD requires reducing stepsizes to achieve linear convergence. To further aid the analysis of our results, we use the "effective passes" as a measure of the number of samples used in the optimization process to the total number of samples in our dataset. This allows us to fix a consistent reference frame for each of the algorithms and allows for the comparison of convergence.

4.3.1 SVRG - Stochastic Variance Reduced Gradient (Three Updates)

To guarantee linear convergence rates for SGD, a decreasing stepsize sequence needs to be used; however, if one wants to use a larger stepsize, a variance reduction method needs to be employed so that the same or better convergence is guaranteed. In order to reduce variance, Johnson and Zhang (2013) introduce SVRG where an estimated $\tilde{w}$ that is close to the optimal w is kept at every iteration and the average gradient $\tilde{\mu}$ is computed. On every *effective pass* of the dataset, the $\tilde{w}$ and $\tilde{\mu}$ is updated and are used for the next iteration. By employing an update in the form of $w^{(t)} = w^{(t-1)} - \alpha(\nabla f_i(w^{(t-1)}) - \nabla f_i(\tilde{w}) + \tilde{\mu})$, where α is the stepsize and i is randomly sampled from $\{1, ..., n\}$, the variance is explicitly reduced. See detail of the algorithm in Algorithm 4.2.

Algorithm 4.2 SVRG Methods for Minimizing an Empirical Risk R_n

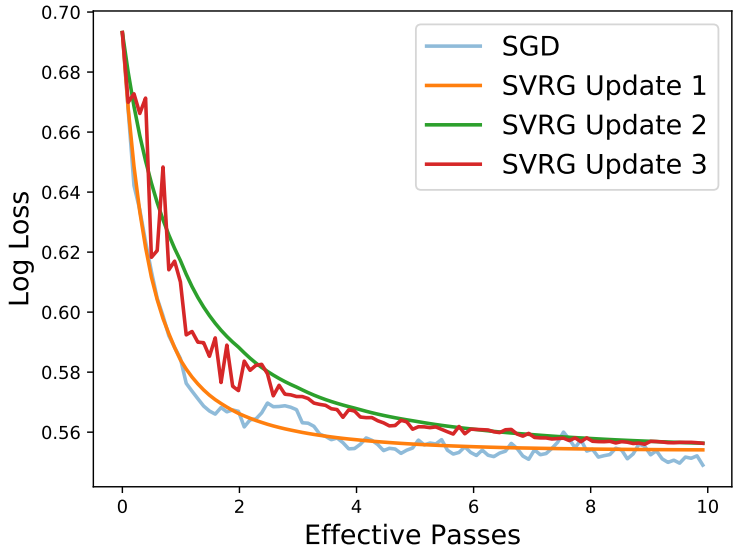
Choose an initial iterate $w_1 \in \mathbb{R}^d$, stepsize $\alpha > 0$, and positive integer m.
for $k = 1, 2, \ldots$ **do**
  Compute the batch gradient $\nabla R_n(w_k)$.
  Initialize $\tilde{w}_1 \leftarrow w_k$.
  for $j = 1, \ldots, m$ **do**
    Chose i_j uniformly from $\{1, \ldots, n\}$.
    Set $\tilde{g}_j \leftarrow \nabla f_{i_j}(\tilde{w}_j) - (\nabla f_{i_j}(w_k) - \nabla R_n(w_k))$.
    Set $\tilde{w}_{j+1} \leftarrow \tilde{w}_j - \alpha \tilde{g}_j$.
  Option (a): Set $w_{k+1} = \tilde{w}_{m+1}$
  Option (b): Set $w_{k+1} = \frac{1}{m} \sum_{j=1}^{m} \tilde{w}_{j+1}$
  Option (c): Choose j uniformly from $\{1, \ldots, m\}$ and set $w_{k+1} = \tilde{w}_{j+1}$.

The SVRG introduces three types of update rule in every iteration ((a), (b), and (c) in Algorithm 4.2). SVRG Update (a) picks the most recent $\tilde{w}_i$ as the update to w. SVRG Update (b) averages the $\tilde{w}_i$ over one effective pass of the data. SVRG Update (c) randomly samples a $\tilde{w}_i$ to use as an update to w.

The authors of Johnson and Zhang (2013) have shown that the algorithm has linear convergence assuming that the objective functions f_i are smooth and convex, and the overall objective function is strongly convex.

In Fig. 4.1, we see that SVRG with updates 2 and 3 converges to the solution with a much smoother descent than SGD. The noise for SVRG Update (c) could be due to the random sampling of the $\tilde{w}_i$. In both SVRG and SGD, we are seeing similar convergence to a solution.

Fig. 4.1: SVRG and SGD



4.3.2 SAGA

SAGA is inspired by SVRG. Instead of approximating the parameters w, SAGA stores the gradient vectors for each sample i. Let $u_i(w_k)$ be the gradient vector for sample i with the weight vector w_k. In each iteration, a random sample j is picked and the weight vector is updated as $w_k = w_{(k-1)} - \alpha \left[\nabla f_j(w_{(k-1)} - u_j + \frac{1}{n}\sum_{i=1}^{n} u_i\right]$. Furthermore, the gradient of sample j is updated as $u_j = \nabla f_j(w_{(k-1)})$. See Algorithm 4.3 for more details.

Algorithm 4.3 SAGA Method for Minimizing an Empirical Risk R_n

Choose an initial iterate $w_1 \in \mathbb{R}^d$ and stepsize $\alpha > 0$.
for $i = 1, \ldots, n$ **do**
  Compute $\nabla f_i(w_1)$.
  Store $\nabla f_i(w_{[i]}) \leftarrow \nabla f_i(w_1)$.
for $k = 1, 2, \ldots$ **do**
  Choose j uniformly in $\{1, \ldots, n\}$.
  Compute $\nabla f_j(w_k)$.
  Set $g_k \leftarrow \nabla f_j(w_k) - \nabla f_j(w_{[j]}) + \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_{[i]})$.
  Store $\nabla f_j(w_{[j]}) \leftarrow \nabla f_j(w_k)$.
  Set $w_{k+1} \leftarrow w_k - \alpha g_k$.

Through storing the full gradient vectors, each iteration corrects the new gradient by using the mean of the gradients through the full sample. The authors of Defazio et al. (2014a) has shown that SAGA has a linear convergence rate similar to that of SVRG with slightly better constants for a specifically chosen step size.

Fig. 4.2: SAGA and SGD

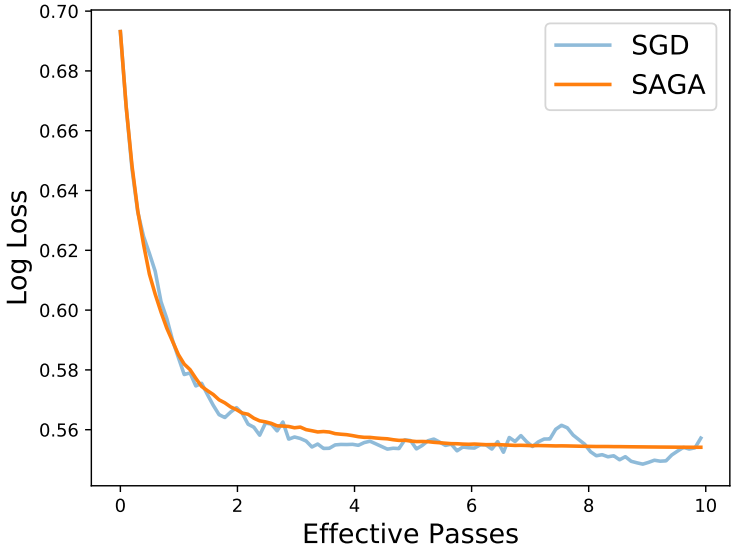


Fig. 4.3: SAGA and SVRG

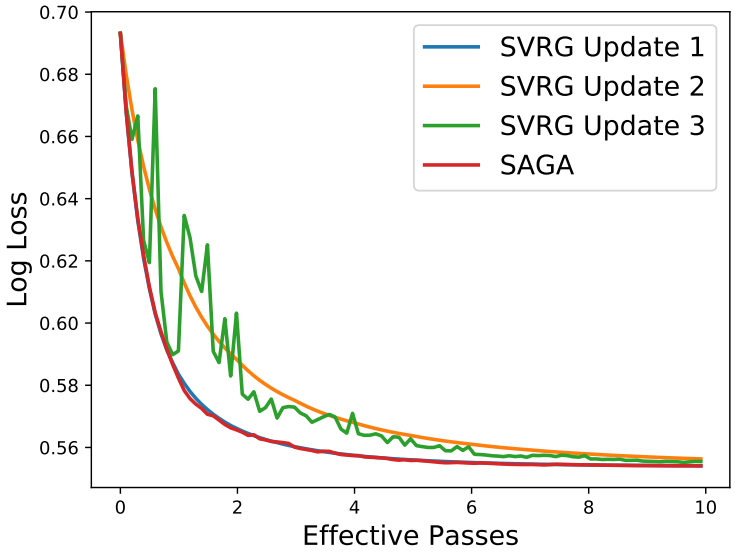


Fig. 4.2 shows SAGA against SGD. We observe that the gradient descent is smooth. Both show a linear convergence rate. Fig. 4.3 shows SAGA against SVRG. We observe that SAGA's performance is very similar to that of SVRG update (a), which uses the most recent $\tilde{w}_i$ estimate for an update.

4.3.3 SAG - Stochastic Average Gradient

SAG is actually the earliest proposed variant of gradient aggregation algorithms we have surveyed. Proposed in 2012 Roux et al. (2012), the algorithm incorporates a memory of previous gradients. This memory allows the algorithm to achieve linear convergence rate with fixed step size. It claims to outperform standard SGD algorithm.

Algorithm 4.4 SAG Method for Minimizing an Empirical Risk R_n

Choose an initial iterate $w_1 \in \mathbb{R}^d$ and stepsize $\alpha > 0$.
for $i = 1, \ldots, n$ **do**
  Compute $\nabla f_i(w_1)$.
  Store $\nabla f_i(w_{[i]}) \leftarrow \nabla f_i(w_1)$.
for $k = 1, 2, \ldots$ **do**
  Choose j uniformly in $\{1, \ldots, n\}$.
  Compute $\nabla f_j(w_k)$.
  Set $g_k \leftarrow \frac{1}{n}\{\nabla f_j(w_k) - \nabla f_j(w_{[j]}) + \sum_{i=1}^{n} \nabla f_i(w_{[i]})\}$.
  Store $\nabla f_j(w_{[j]}) \leftarrow \nabla f_j(w_k)$.
  Set $w_{k+1} \leftarrow w_k - \alpha g_k$.

Under a strongly convex sum of smooth functions, the SAG algorithm has linear convergence. The difference between SAG and SAGA is that SAG's updates the weight vectors as $w_k = w_{(k-1)} - \alpha \frac{1}{n}\left[\nabla f_j(w_{(k-1)} - u_j + \sum_{i=1}^{n} u_i\right]$. The disadvantage of SAG is that it uses a biased estimator of the gradient unlike SAGA as detailed by Defazio et al. (2014a). Details are shown in Algorithm 4.4.

Fig. 4.4: SAG and SVRG

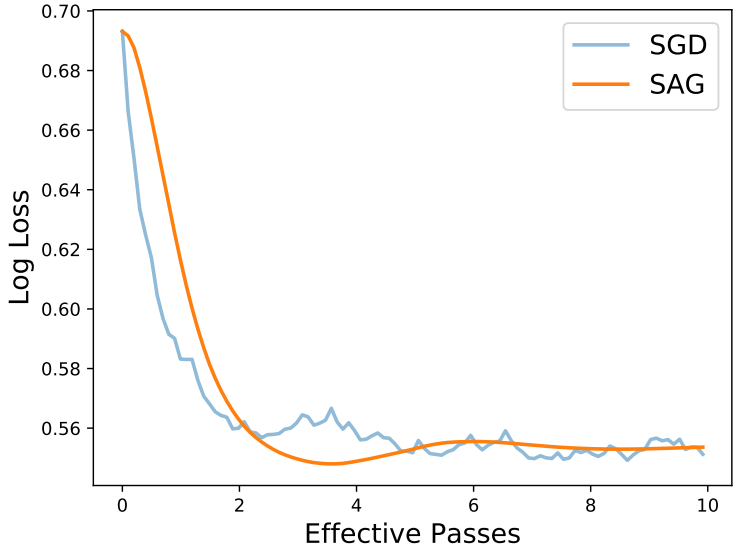


Fig. 4.5: SAG and SAGA

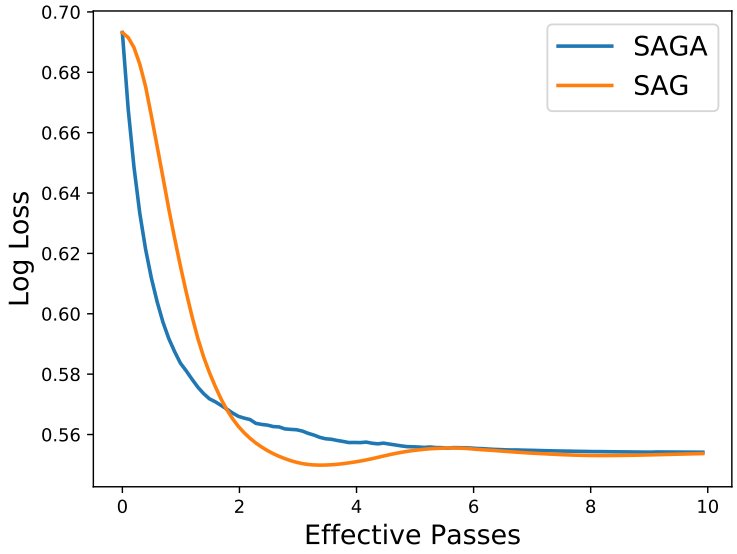


Fig. 4.4 shows SAG against SGD. We observe a linear convergence of SAG similar to that of SGD. The descent of SAG is also smooth unlike SGD. However, the convergence displays a bit of a wiggle. This was consistent throughout the many runs we conducted. Fig. 4.5 shows the comparison between SAG and SAGA.

4.3.4 S2GD - Semi-Stochastic Gradient Descent

Algorithm 4.5 S2GD Method for Minimizing an Empirical Risk R_n

Choose an initial iterate $w_1 \in \mathbb{R}^d$, stepsize $\alpha > 0, \nu > 0, \beta > 0$, and positive integer m.
for $k = 1, 2, \ldots$ **do**
  Compute the batch gradient $\nabla R_n(w_k)$.
  Initialize $\tilde{w}_1 \leftarrow w_k$.
  for $j = 1, \ldots, m$ **do**
    Chose i_j uniformly from $\{1, \ldots, n\}$.
    Set $\tilde{g}_j \leftarrow \nabla f_{i_j}(\tilde{w}_j) - (\nabla f_{i_j}(w_k) - \nabla R_n(w_k))$.
    Set $\tilde{w}_{j+1} \leftarrow \tilde{w}_j - \alpha \tilde{g}_j$.
  Choose j from $\{1, \ldots, m\}$ with probability $\frac{1}{\beta}(1 - \nu\alpha)^{m-t}$ for $t \in [m]$.
  Set $w_{k+1} = \tilde{w}_{j+1}$.

In SVRG, Johnson and Zhang (2013) named three update methods. One of the update methods picks w uniformly at random. S2GD imposes a distribution on the update instead of a uniform distribution. The recent weights updates get more

probability mass than the older one. Thus, SVRG then is a special case of S2GD. Detail is shown in Algorithm 4.5.

The convergence rate of S2GD is the same as that of SVRG with differences in constants. Both exhibit linear convergence. However, Konen and Richtrik (2017) showed that S2GD+, a variant of S2GD without theoretical analysis, shows better convergence in practice.

Fig. 4.6: S2GD and SGD

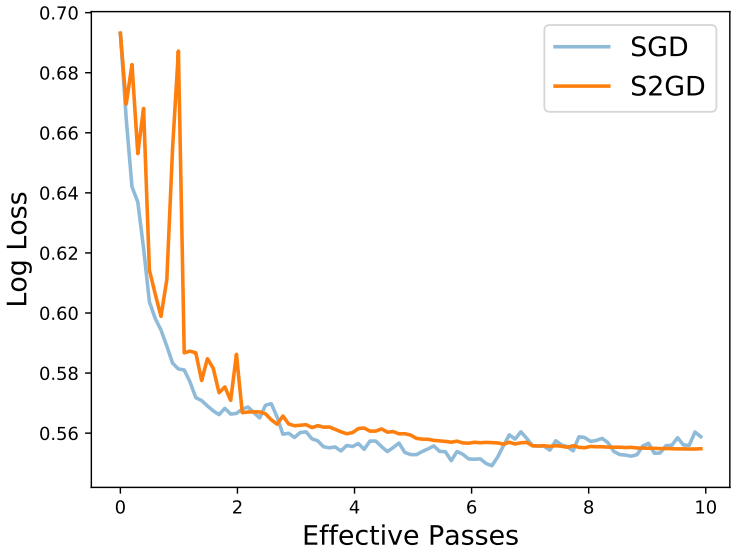


Fig. 4.7: S2GD and SVRG

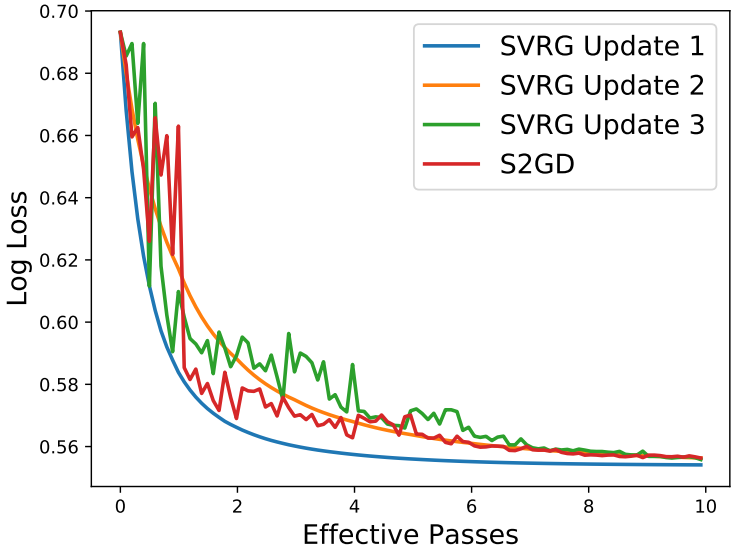


Fig. 4.6 shows S2GD and SGD. S2GD showed large variance in the early passes but the variance reduces as the algorithm converges to the solution. Fig. 4.7 shows SVRG and S2GD. We can see that S2GD shows similar convergence to SVRG Update (c), which is the one where updates are picked uniformly.

4.3.5 Finito - Faster, Permutable Incremental Gradient

As opposed to the other methods we have just surveyed, Finito method samples the data without replacement during iterations. The authors Defazio et al. (2014b) claim that sampling without replacement is better for much faster convergence. For each iteration, the updates are similar to that of SAGA where the gradient vectors for each sample is stored and similar to SVRG where the approximation of w is also stored. For each iteration, w is updated using the average of the approximations of w along with the average of the gradients. Details are shown in Algorithm 4.6.

Algorithm 4.6 Finito Method for Minimizing an Empirical Risk R_n

Choose initial iterates $w_1, ..., w_n \in \mathbb{R}^d$ and stepsize $\alpha > 0$.
for $i = 1, \ldots, n$ **do**
  Compute $\nabla f_i(w_i)$.
  Store $\nabla f_i(w_{[i]}) \leftarrow \nabla f_i(w_i)$.
  Store $w_{[i]} \leftarrow w_i$.
for $k = 1, 2, \ldots$ **do**
  Choose j uniformly without replacement in $\{1, \ldots, n\}$ (Set array $\{1, ..., n\}$ if $k > n$ for sampling).
  Set $\bar{g}_k \leftarrow \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_{[i]})$.
  Set $w_k \leftarrow \frac{1}{n} \sum_{i=1}^{n} w_{[i]} - \alpha \bar{g}_k$.
  Store $w_{[j]} \leftarrow w_k$.
  Compute $g_k \leftarrow \nabla f_j(w_k)$.
  Store $\nabla f_j(w_{[j]}) \leftarrow \nabla f_j(w_k)$.

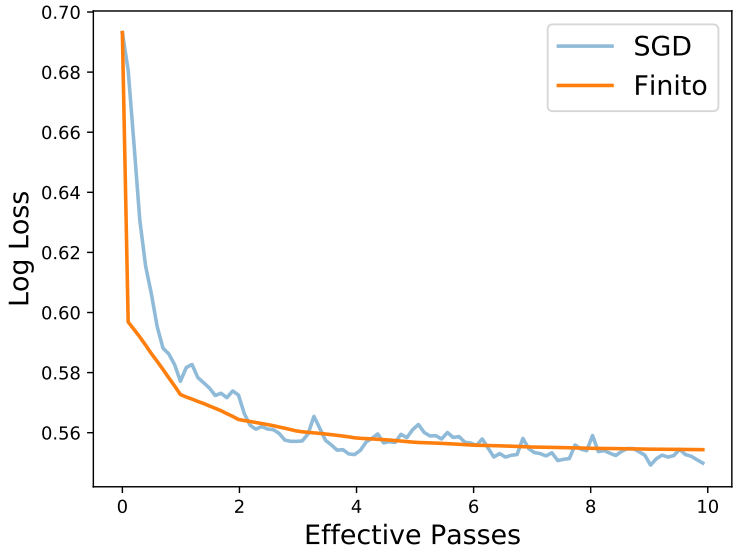
Finito exhibits linear convergence rate similar to that of SGD with reducing step size given that the objective function is Lipschitz continuous and strongly convex.

Fig. 4.8 shows Finito and SGD. Finito shows a smooth descent to the solution. This is expected given that in each iteration, only one gradient and approximation to w is updated, generating minor effect on the overall descent when updating with the average over the entire dataset.

4.3.6 VR-Lite

VR-Lite De et al. (2015) is proposed as a variance reduced SGD variant to reduce the high memory usage in examples such as Finito and large batch gradient computations in S2GD. VR-Lite performs SGD for the first effective pass of the data to initialize the w vectors and the iteration averages $\bar{w}$ and $\bar{g}$. VR-Lite then updates w by going

Fig. 4.8: Finito and SGD



through the samples without replacement and updating the parameters by using $\bar{w}$ and $\bar{g}$ as correction terms. See detail in Algorithm 4.7.

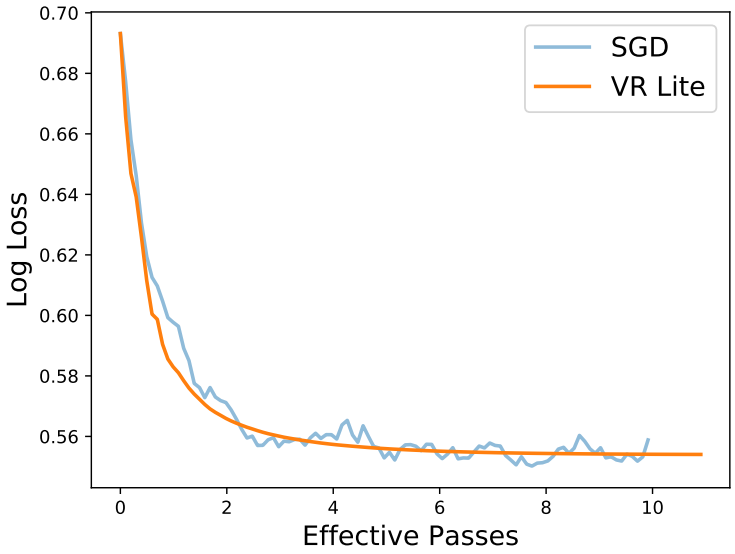
Algorithm 4.7 VR-Lite Method for Minimizing an Empirical Risk R_n

Choose initial iterates $w \in \mathbb{R}^d$ and stepsize $\alpha > 0$.
Initialize $\bar{w} = w$ and $\bar{g} = \nabla f(w)$.
for $i = 1, \ldots, n$ **do**
  Sample i uniformly in $\{1, ..., n\}$.
  Compute $\nabla f_i(w)$.
  Set $w \leftarrow w - \alpha \nabla f_i(w)$.
  Set $\bar{w} = \bar{w} + w$.
  Set $\bar{g} = \bar{g} + \nabla f_i(w)$.
Set $\bar{w} = \frac{1}{n}\bar{w}$.
Set $\bar{g} = \frac{1}{n}\bar{g}$.
for $k = 1, 2, \ldots$ **do**
  Initialize $\tilde{w} = \tilde{g} = 0$.
  for $j = 1, \ldots, n$ **do**
    Chose i_j uniformly from $\{1, \ldots, n\}$ without replacement.
    Set $w \leftarrow w - \alpha(\nabla f_{i_j}(w) - \nabla f_{i_j}(\bar{w}) + \bar{g})$.
    Set $\tilde{w} = \tilde{w} + w$.
    Set $\tilde{g} = \tilde{g} + \nabla f_{i_j}(w)$.
  Set $\bar{w} = \frac{1}{n}\tilde{w}$.
  Set $\bar{g} = \frac{1}{n}\tilde{g}$.

VR-Lite preserves linear convergence rates under strongly convex functions. However, the author noted that although convergence can be shown experimentally, proving it theoretically is difficult.

Fig. 4.9 shows VR-Lite and SGD. We can see that during the first effective pass, the convergence is exactly the same as SGD. However near the optimum, the VR-Lite is smoother due to the extra correction terms during each update.

Fig. 4.9: VR-Lite and SGD



4.3.7 Batching and Mixed SVRG

Batching SVRG Harikandeh et al. (2015) differs from SVRG in the sense that instead of sampling over the entire dataset, a batch is chosen at every iteration. The samples are then drawn without replacement from the batch. Each update is then similar to that of SVRG where the gradient of the entire batch is added as a correction term. The batch sizes can be dynamically sized from iteration to iteration.

Mixed SVRG Harikandeh et al. (2015) is a hybrid between SGD and Batching SVRG. The only difference between Batching SVRG and Mixed SVRG is during the sampling. If the sample chosen is not in the batch, then normal stochastic gradient is calculated and used as an update. If the chosen sample is in the batch, then an SVRG update is done. Detail of Batching SVRG is shown in Algorithm 4.8 and Mixed SVRG in Algorithm 4.9.

The authors of Harikandeh et al. (2015) show proof for linear convergence rates for Batching and Mixed SVRG for strongly convex functions. For Mixed SVRG, a reducing stepsize is still required for the stochastic gradient part just like SGD to guarantee the linear convergence.

Algorithm 4.8 Batching SVRG Methods for Minimizing an Empirical Risk R_n

Choose an initial iterate $w_1 \in \mathbb{R}^d$, stepsize $\alpha > 0$, and positive integers B and m.

for $k = 1, 2, \ldots$ **do**
  Sample a batch of size B without replacement from $\{1, ..., n\}$.
  Compute the batch gradient $\nabla R_B(w_k) = \sum_{i \in B} \nabla f_i(w_k)$.
  Initialize $\tilde{w}_1 \leftarrow w_k$.
  for $j = 1, \ldots, m$ **do**
    Chose i_j uniformly from $\{1, \ldots, n\}$.
    Set $\tilde{g}_j \leftarrow \nabla f_{i_j}(\tilde{w}_j) - (\nabla f_{i_j}(w_k) - \nabla R_B(w_k))$.
    Set $\tilde{w}_{j+1} \leftarrow \tilde{w}_j - \alpha \tilde{g}_j$.
  Choose j uniformly from $\{1, \ldots, m\}$ and set $w_{k+1} = \tilde{w}_{j+1}$.

Algorithm 4.9 Mixed SVRG Methods for Minimizing an Empirical Risk R_n

Choose an initial iterate $w_1 \in \mathbb{R}^d$, stepsize $\alpha > 0$, and positive integers B and m.

for $k = 1, 2, \ldots$ **do**
  Sample a batch of size B without replacement from $\{1, ..., n\}$.
  Compute the batch gradient $\nabla R_B(w_k) = \sum_{i \in B} \nabla f_i(w_k)$.
  Initialize $\tilde{w}_1 \leftarrow w_k$.
  for $j = 1, \ldots, m$ **do**
    Chose i_j uniformly from $\{1, \ldots, n\}$.
    if $i_j \in B$ **then**
      Set $\tilde{g}_j \leftarrow \nabla f_{i_j}(\tilde{w}_j) - (\nabla f_{i_j}(w_k) - \nabla R_B(w_k))$.
    else
      Set $\tilde{g}_j \leftarrow \nabla f_{i_j}(\tilde{w}_j)$.
    Set $\tilde{g}_j \leftarrow \nabla f_{i_j}(\tilde{w}_j) - (\nabla f_{i_j}(w_k) - \nabla R_B(w_k))$.
    Set $\tilde{w}_{j+1} \leftarrow \tilde{w}_j - \alpha \tilde{g}_j$.
  Choose j uniformly from $\{1, \ldots, m\}$ and set $w_{k+1} = \tilde{w}_{j+1}$.

Fig. 4.10 shows Batching SVRG with SGD. Fig. 4.11 shows Mixed SVRG with SGD. Both of them show linear convergence. We can see that in Mixed SVRG, there is higher variance near the convergence to the solution like that of SGD. This is due to the hybrid nature of Mixed SVRG where some SGD steps are taken instead of SVRG.

Fig. 4.10: Batching SVRG and SGD

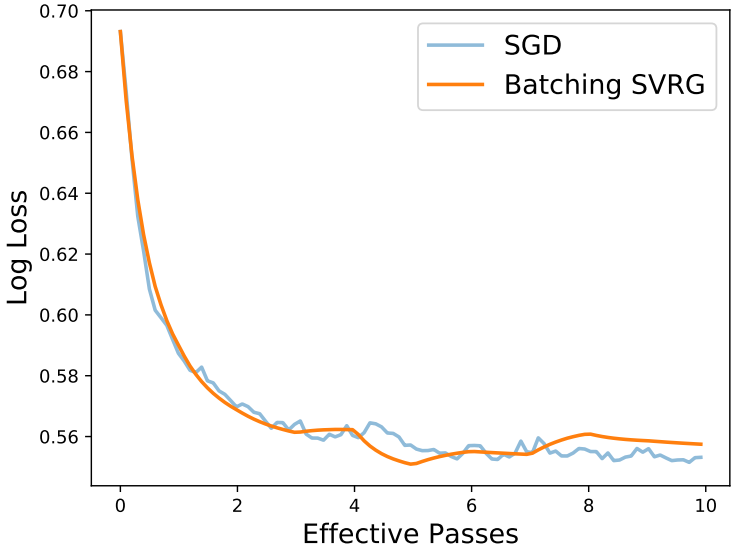
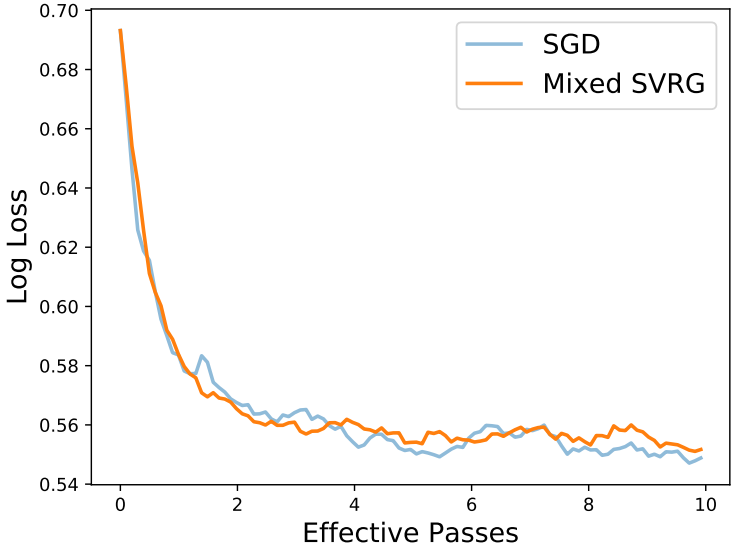


Fig. 4.11: Mixed SVRG and SGD



4.4 Results on Subset 0/1 Loss

We applied the various variants of the gradient aggregation methods on the Subset 0/1 Loss problem as described in 4.2. We selected the Label Powerset 3.3 approach to reduce the multilabel classification problem to a multiclass classification problem. We identified the 27 unique labels and converted them into one hot encoding vectors. Using the cross entropy loss as our surrogate loss for Subset 0/1 Loss, we formulate a convex optimization problem that is a sum of convex and smooth functions.

Fig. 4.12: Subset 0/1 Loss on Various Gradient Aggregation methods

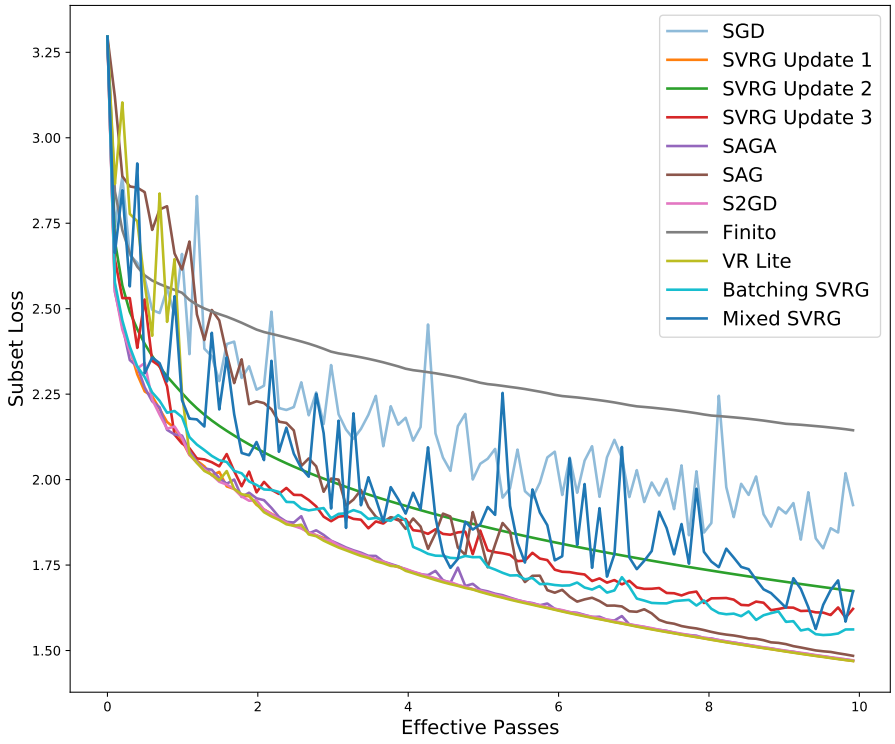


Fig. 4.12 shows the results of the optimization using various gradient aggregation methods. We observe that SGD and Finito suffer the most. Although, Finito has smoother curve than SGD. SGD suffered from high variance during the gradient descent process while the other gradient aggregation strategies were able to converge with lower variance. The other methods were also able to converge to a solution with lower subset loss compared to SGD due to the gradient correction terms. All these plots are run with similar parameters to ensure that this comparison is valid. This behavior can be attributed to the fact that minimizing subset loss is much harder problem than minimizing Hamming loss, and thus more sophisticated variance reduction methods are performing better than usual SGD.

5. Conclusion

Stochastic Gradient Descent (SGD) methods have become popular these days and are being implemented by every major organization in academia as well as industry. As the data is increasing day by day, it is important to understand these optimization

methods for large scale learning. In this project, we have tried to deep dive into these algorithms and provide a coherent summarized view from the implementation perspective. Further, in order to see these optimization methods in action, we study two loss functions - Hamming Loss and Subset 0/1 loss along with their similarities and differences. These two losses identify two approaches to the multi-label problem - reduction to independent binary classification problems and reduction to a multi-class classification problem. The report contains algorithms for ten different variants of SGD for two different loss functions along with their results on a benchmark dataset.

We hope that not only this report would be useful for us in our future research work in Machine Learning and Computational Inference, but also would be useful for beginners' who want to understand these variants of SGD or who want to explore Multi-label classification.

References

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

S. De, G. Taylor, and T. Goldstein. Variance Reduction for Distributed Stochastic Gradient Descent. *ArXiv e-prints*, December 2015.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1646–1654. Curran Associates, Inc., 2014a.

Aaron Defazio, Justin Domke, and Tibério S. Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In *ICML*, 2014b.

Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45, 2012.

Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečný, and Scott Sallinen. Stopwasting my gradients: Practical svrg. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2251–2259. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5711-stopwasting-my-gradients-practical-svrg.pdf.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 315–323,

USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999611.2999647.

Jakub Konen and Peter Richtrik. Semi-stochastic gradient descent methods. *Frontiers in Applied Mathematics and Statistics*, 3:9, 2017. ISSN 2297-4687. doi: 10.3389/fams.2017.00009. URL https://www.frontiersin.org/article/10.3389/fams.2017.00009.

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

Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2663–2671, USA, 2012. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999325.2999432.

Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, volume 8, pages 325–330, 2008.

Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis Vlahavas. Multi-label classification of music by emotion. *EURASIP Journal on Audio, Speech, and Music Processing*, 2011(1):4, Sep 2011. ISSN 1687-4722. doi: 10.1186/1687-4722-2011-426793. URL https://doi.org/10.1186/1687-4722-2011-426793.

George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

Dan Yang and Won-Sook Lee. Disambiguating music emotion using software agents. In *ISMIR*, volume 4, pages 218–223, 2004.