Low-Cost Lipschitz-Independent Adaptive Importance Sampling of Stochastic Gradients

Huikang Liu, Xiaolu Wang, Jiajin Li, Anthony Man-Cho So
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Hong Kong SAR, China
Email: {hkliu, xlwang, jjli, manchoso}@se.cuhk.edu.hk

***Abstract*—Stochastic gradient descent (SGD) usually samples training data based on the uniform distribution, which may not be a good choice because of the high variance of its stochastic gradient. Thus, importance sampling methods are considered in the literature to improve the performance. Most previous work on SGD-based methods with importance sampling requires the knowledge of Lipschitz constants of all component gradients, which are in general difficult to estimate. In this paper, we study an adaptive importance sampling method for common SGD-based methods by exploiting the local first-order information without knowing any Lipschitz constants. In particular, we periodically changes the sampling distribution by only utilizing the gradient norms in the past few iterations. We prove that our adaptive importance sampling non-asymptotically reduces the variance of the stochastic gradients in SGD, and thus better convergence bounds than that for vanilla SGD can be obtained. We extend this sampling method to several other widely used stochastic gradient algorithms including SGD with momentum and ADAM. Experiments on common convex learning problems and deep neural networks illustrate notably enhanced performance using the adaptive sampling strategy.**

I. Introduction

An extensively studied problem in machine learning and pattern recognition is that of empirical risk minimization (ERM), i.e.,

$$\min_{\mathbf{w}\in\mathbb{R}^d} \left\{ F(\mathbf{w}) \coloneqq \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{w}) \right\}, \tag{1}$$

where $f_i : \mathbb{R}^d \to \mathbb{R}$ is the i-th component function. In general, $f_i(\mathbf{w}) = \ell(h(\mathbf{x}_i;\mathbf{w}),\mathbf{y}_i)$, where $(\mathbf{x}_i,\mathbf{y}_i) \in \mathcal{X}\times\mathcal{Y}$ is the i-th training example, h is the decision function parameterized by $\mathbf{w}$, and ℓ is the loss function. SGD, which dates back to the original work by [30], is playing a central role in solving optimization problem (1) [4]. In each iteration of vanilla SGD, the full gradient $\nabla F(\mathbf{w}_k)$ is approximated by a randomly sampled stochastic gradient $\nabla f_i(\mathbf{w}_k)$, where i follows the uniform distribution.

The common uniform sampling endows the stocahstic gradients with high variance, which has negative influence on the convergence rate. Therefore, importance sampling is brought to stochastic optimization algorithms, where more delicate sampling probabilities are assigned to different training examples. The existing work on importance sampling mostly utilizes the Lipschitz constants of f_i or ∇f_i to assign each training sample with a proper sampling probability. For example, the reweighted sampling probability in [19] and [40] is $p_i = L_i/\sum_{j=1}^{n} L_j$ for all $i \in [n] \coloneqq \{1,2,\ldots,n\}$, where L_i is the Lipschitz constant of ∇f_i. In most cases, the Lipschitz constant of a function is not easy to obtain or even estimate. Moreover, if a stationary importance sampling distribution is adopted throughout the learning process, only the global property of the objective function can be employed.

In this paper, we introduce an efficient adaptive importance sampling methodology that can be applied to different algorithms based on stocahstic gradients. We first consider to incorporate this adaptive importance sampling idea into vanilla SGD, namely SGD-AIS. Unlike those importance sampling schemes that rely on the information of Lipschitz constants, SGD-AIS are only based on the component gradient norms in the past few iterations. For SGD-AIS, little extra computation (i.e., extra $O(\log(n))$ per-iteration computation), as well as minor storage cost (i.e., $O(n)$ extra space requirement) are introduced. Furthermore, instead of fixing the sampling distribution on top of the global function property during the entire optimization, SGD-AIS adjusts the sampling distribution every iteration. This allows the importance sampling to constantly capture the local information of the objective function and can better approximate the optimal sampling distribution mentioned in [41]. To avoid taking excessive time in computing the adaptive sampling distributions, we employ a binary tree structure to assist storing and updating the varying probabilities. With moderate storage requirement, we can achieve very effiecient update and sampling by visiting the binary tree. Our theoretical analysis suggests that under mild conditions, SGD-AIS notably reduces the variance of the stochastic gradient in an non-asymptotic sense. Thereby, compared with vanilla SGD, the convergence bounds can be further improved in both convex and nonconvex setting. We provide a novel proof of the main theoretical result by constructing an auxiliary sampling distribution. Furthermore, we extend this adaptive importance sampling to SGD with momentum and ADAM, which are also computationally efficient and Lipschitz-independent. Experimental results on several datasets exhibit much faster convergence for all our adaptive sampling-based algorithms, when they are applied to logistic regression, support vector machine and deep neural networks.

A. Related Works

The goal of solving roblem 1 is to obtain an ϵ-optimal solution $\hat{x}$. Full gradient descent requires $O((1+\kappa)\log(1/\epsilon))$

iterations [21], [23]. If Nesterov's momentum is further imported, an improved $O((1+\sqrt{\kappa})\log(1/\epsilon))$ iteration complexity can be achieved [3], [38]. However, full gradient involves the whole dataset every iteration, with excessively high $O(nd)$ computational cost. As a comparison, SGD has dominated large-scale learning as well as deep learning [2], [39], [9], because of it being both well understood in theory and effective in practice. SGD randomly selects only one data sample or a mini batch of data samples at each iteration, which takes only $O(d)$ per-iteration cost, but higher $O(\kappa/\epsilon)$ overall iterations [5]. For a general convex and L-smooth objective function, SGD achieves a convergence rate of $O(1/\epsilon^2)$ [32]. When F is strongly convex and L-smooth, SGD converges to the global minimum at a rate of $O(1/\epsilon)$ [20], [33]. For a nonconvex L-smooth objective function, SGD converges to a first-order stationary point at a rate of $O(1/\epsilon^2)$ in general, but faster rate of $O(1/\epsilon)$ can be obtained if the objective function further satisfies the so-called Polyak-Łojasiewicz condition [12], [16].

To further reduce the variance of the stochastic gradient, many different gradient aggregation techniques are proposed. Some representatives include SAG [31], SVRG [15], SAGA [8], and SARAH [24]. SAGA iteratively computes a stochastic vector $\mathbf{g}_k$ as the average of stochastic gradients evaluated at previous iterates. SVRG keeps track of a "snapshot" vector $\mathbf{w}_k$ that is updated once every a fixed number of iterations, and computes full gradient only for the snapshot vector. In general, these variance reduction methods can achieve linear convergence given that the objective function is strongly convex and has Lipschitz continuous gradient [35]. Spider [11], which achieve comparable $O((n+\kappa)\log(1/\epsilon))$ iteration complexity for strongly convex loss functions. An accelerated variance-reduced optimization algorithm, named Katyusha, is proposed in [1], which improves the iteration complexity to $O((n+\sqrt{n\kappa})\log(1/\epsilon))$. However, they either periodically compute full gradient or store all n component gradients, which makes them less used in large-scale scenarios including training deep neural networks.

Another important line of research is to incorporate SGD with the first and second moment estimates which computes individual adaptive stepsizes for different coordinates. SGD with momentum computes the stochastic gradient based on the exponential moving averge of the historical gradients. AdaGrad [10] uses the second moment estimates to help find the very predictive but rarely seen features. The second moment in AdaGrad accumulates and is monotonically increasing, which may lead to the tranning stopping too early. RMSProp [37] overcomes this by using the expomential moveing average of the second moments. ADAM [17], [28] combines the advantages of the first and second moment estimates, and has been rather popular in deep learning.

Importance sampling is often involved in picking the stochastic gradients and random coordiantes [19], [40], [41], [27], [13], [14], where an appropriately chosen non-uniform stationary sampling distribution is used. Such stationary sampling usually requires some sorts of prior knowledge like component-wise or coordianate-wise Lipschitz constants. Adaptive importance sampling methods [7], [25], [34], [26], [36] are further considered to explore more local function information, but a higher computational cost is usually needed because of the frequent change of sampling distributions. For instance, AdaSDCA [7] adapts the coordinate sampling probabilities based on the dual formulation of ERM. However, this method requires to evaluate all the component dual residuals every iteration, which makes it less computationally practical. In comparison, our work takes advantage of adaptive importance sampling while introducing little extra computational cost.

II. Adaptive Importance Sampling

Applying vanilla SGD to solve optimization problem (1), the following update rule of the model parameter $\mathbf{w}$ in the k-th step is performed

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k \nabla f_{i_k}(\mathbf{w}_k), \tag{2}$$

where η_k is the stepsize and i_k is uniformly chosen in $[n]$. Obviously we have $\mathbb{E}[\nabla f_{i_k}(\mathbf{w}_k)] = \nabla F(\mathbf{w}_k)$, meaning that the stochastic gradient is unbiased. However, the variance of the stochastic gradient has not been controlled, which may negatively affect the convergence performance. To obtain a stochastic gradient with smaller variance, we employ certain importance sampling distribution $\boldsymbol{p}^k := (p_1^k, p_2^k, \ldots, p_n^k)^\top$ in the k-th iteration, where p_i^k stands for the probability that the i-th training example is picked. Thereupon, the stochastic gradient in the k-th iteration with importance sampling distribution $\boldsymbol{p}^k$ should be

$$\mathbf{g}_k = \frac{1}{np_{i_k}^k}\nabla f_{i_k}(\mathbf{w}_k), \tag{3}$$

where the multiplied factor $1/np_{i_k}^k$ is to make sure that $\mathbf{g}_k$ is an unbiased estimator of the gradient. The model parameter is further moved along the direction of negative stochastic gradient:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k \mathbf{g}_k. \tag{4}$$

A natural idea of choosing distribution $\boldsymbol{p}^k$ is to find the one that minimizes the variance of $\mathbf{g}_k$, or equivalently, solves the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{p}^k} \quad & \mathrm{Var}\left[\frac{1}{np_i^k}\nabla f_i(\mathbf{w}_k)\right] \\ \text{s. t.} \quad & 0 \le p_i^k \le 1,\ \forall i \in \{1,\ldots,n\} \\ & \sum_{j=1}^n p_j^k = 1. \end{aligned} \tag{5}$$

Problem (5) has a closed-form optimal solution [41], which is

$$(p_i^k)^* = \frac{\|\nabla f_i(\mathbf{w}_k)\|_2}{\sum_{j=1}^n \|\nabla f_j(\mathbf{w}_k)\|_2},\ \forall i \in \{1,\ldots,n\}. \tag{6}$$

This optimal sampling method minimizes the varaince by fully exploiting the local information in every iterate $\mathbf{w}_k$. However, it is impractical because it requires calculation of all n component gradient norms every iteration, so that it reaches the same computational complexity as the full gradient method. Thus,

[41] adopts a more computationally feasible distribution in their algorithm by replacing each gradient norm with its global upper bound. However, such relaxed distribution involves the Lipschitz constants of f_i or ∇f_i, just like many other importance sampling-based algorithms. Instead, we will show that the gradient norms in (6) that are evaluated at some iterate $\mathbf{w}_k$ can be approximated by other local information, i.e., the most recently evaluated gradient norms. This will significantly mitigate the per-iteration computational cost of (6), at the expense of some sort of inexactness.

A. SGD-AIS

Motivated by the above discussion, an importance sampling approach for SGD, named SGD-AIS, is introduced for solving problem (1), whose algorithmic details are shown in Algorithm 1. In the k-th iteration of SGD-AIS, the sampling distribution $\boldsymbol{p}$ is fixed to be a mixture of a reweighted sampling distribution $\mathcal{G}_k$ and the uniform distribution $\mathcal{U}$. The probability for sampling the i-th training example is

$$p_i^k = \alpha_k p_i^{\mathcal{G}_k} + (1-\alpha_k) p_i^{\mathcal{U}},\ \forall i \in [n], \tag{7}$$

where $p_i^{\mathcal{G}_k} = \pi_i / \sum_{j=1}^n \pi_j$ and $p_i^{\mathcal{U}} = 1/n$ for $i \in [n]$. The parameter $\alpha_k \in (0,1)$ is for controlling the proportions of these two distributions. We let α_k be both lower and upper bounded, and increase with k. Hence, at the initial stage of the algorithm, α_k is small, thus uniform sampling $\mathcal{U}$ contributes more to the overall sampling distribution. As the learning process goes on, the overall sampling distribution will more and more approach the reweighted sampling $p^{\mathcal{G}_k}$.

Algorithm 1 SGD-AIS

1: **Input:** step sizes $\{\eta_k\}$, weights $\alpha_k \in [\underline{\alpha}, \overline{\alpha}] \subseteq (0,1)$ for all $k \in \mathbb{N}$.
2: **Initialize:** $\mathbf{w}_1$, $\pi_i = 1$ for all $i \in \{1,\dots,n\}$
3: **for** $k = 1,,2\dots$ **do**
4: Update the sampling probabilities for all $i \in \{1,\dots,n\}$

$$p_i = \alpha_k \frac{\pi_i}{\sum_{j=1}^n \pi_j} + (1-\alpha_k)\frac{1}{n} \tag{8}$$

5: Randomly pick $i_k \in [n]$ based on distribution $\boldsymbol{p}$
6: Compute stochastic gradient

$$\mathbf{g}_k = \frac{1}{n p_{i_k}} \nabla f_{i_k}(\mathbf{w}_k) \tag{9}$$

7: Set $\pi_{i_k} = \|\nabla f_{i_k}(\mathbf{w}_k)\|_2$
8: Set $\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k \mathbf{g}_k$
9: **end for**

The distribution $\mathcal{G}$ is given in the form of $p_i^{\mathcal{G}} = \|\nabla f(\mathbf{w}_{\tau_i})\|_2 / \sum_{j=1}^n \|\nabla f(\mathbf{w}_{\tau_j})\|_2$ for all $i \in [n]$, which has a similar form as the optimal distribution given in (6). Its main difference from (6) is that the gradient norms in $p_i^{\mathcal{G}}$ are evaluated at different iterates. In other words, $\mathcal{G}$ uses "inexact" gradient information evaluated at n different past iterates to approximate the "exact" gradient information that are all evaluated at the current iterate. Despite the sub-optimality of our sampling distribution, we can show that it reduces the variance of the stochastic gradient as compared to the uniform distribution, whose details are given in the next subsection.

The adaptivity of the importance sampling scheme guarantees that the gradient information that are collected is close to the current iteration. As such, without using any generally unknown Lipschitz constants, SGD-AIS engages the approximate local structure of the objective function into its importance sampling, yet has the same order of per-iteration computational cost with vanilla SGD.

B. SGDm-AIS

According to the framework of SGD-AIS, we can also apply this adaptive importance sampling method to other methods based on stochastic gradients. Incorporated with momentum-based SGD results in the algorithm that we call SGDm-AIS. Different from the stochastic gradient (9) of SGD-AIS, the stochastic gradient of SGDm-AIS is defined is recursively, that is

$$\mathbf{g}_k = \theta \mathbf{g}_{k-1} + (1-\theta)\frac{1}{n p_{i_k}^k} \nabla f_{i_k}(\mathbf{w}_k), \tag{10}$$

where θ and $1-\theta$ weight the historical gradient and current gradient. Different from standard SGD with momentum, in SGDm-AIS we involve the newly evaluated gradient by multiplying a factor of $1/(np_{i_k}^k)$ to make it unbiased.

C. ADAM-AIS

As for ADAM-AIS, the stochastic gradient is defined as

$$\mathbf{g}_k = \frac{\hat{\mathbf{m}}_k}{\sqrt{\hat{\mathbf{h}}_k} + \varepsilon}, \tag{11}$$

where

$$\hat{\mathbf{m}}_k = \left(\theta_1 \mathbf{m}_{k-1} + (1-\theta_1)\frac{1}{n p_{i_k}^k}\mathbf{g}_k\right) \Big/ (1-\theta_1^k), \tag{12}$$

and

$$\hat{\mathbf{h}}_k = \left(\theta_2 \mathbf{h}_{k-1} + (1-\theta_2)\frac{1}{n p_{i_k}^k}\mathbf{g}_k^2\right) \Big/ (1-\theta_2^k). \tag{13}$$

It is noted that, to make them unbiased, the newly evaluated gradient and squared gradient are both multiplied by an $1/(np_{i_k}^k)$ factor.

Regarding the the adaptive importance sampling as a type of methodology, we can also apply it to many other stochastic optimization algorithms, like SVRG, SAGA, AdaGrad, RMSProp, etc.

D. Mini-Batch Variants

SGD-AIS (as well as SGDm-AIS and ADAM-AIS) can be naturally extended to mini-batch cases. The mini-batch SGD-AIS selects more than one component function to compute the gradient. Let m be the mini-batch size. A simple way is to partition the set of indices into n/m disjoint subsets $\{I_1,\dots,I_{n/m}\}$, so that the cardinality of each subset is equal to m (assume n is a multiple of m). Then, during each

iteration, randomly select one subset based on the distribution $\boldsymbol{p}^k$, where the corresponding optimal distribution is given by

$$(p_i^k)^* = \frac{\|\sum_{l\in I_i}\nabla f_l(\mathbf{w}_k)\|_2}{\sum_{j=1}^{n/m}\|\sum_{l\in I_j}\nabla f_l(\mathbf{w}_k)\|_2},\ \forall i\in\{1,\ldots,n/m\}. \tag{14}$$

However, it is not a flexible way because the elements of each mini-batch are fixed. We implement the mini-batch variants in another way [29]. Firstly, we partition the set of indices into m disjoint subsets $\{J_1,\ldots,J_m\}$, so that the cardinality of each subset is equal to n/m. Then, during each iteration, randomly select a single index from each subset and add it to the mini-batch. We perform the adaptive importance sampling on each individual subset. So the optimal distribution for each subset J_s $(s=1,\ldots,m)$ is given by

$$(p_i^k)^* = \frac{\|\nabla f_i(\mathbf{w}_k)\|_2}{\sum_{j\in J_s}\|\nabla f_j(\mathbf{w}_k)\|_2},\ \forall i\in J_s. \tag{15}$$

E. Efficient Implementation of Sampling

In SGD-AIS, SGDm-AIS or ADAM-AIS, we need to pick an index that follows non-uniform distribution p^k at every iteration. A natural way to implement this is to first generate a uniformly distributed random number, then perform binary search based on the sorted array $[p_1^k, p_1^k+p_2^k,\ldots,\sum_{j=1}^n p_j^k]$. This results in only $O(\log n)$ sampling time [22]. However, some π_i will be changed every iteration, thus updating the array costs $O(n)$ time, in addtional to the $O(d)$ per-iteration cost of computing a stochastic gradient. To mitigate the frequent update of probability distribution, we resort to a binary tree data structure $\mathcal{T}$. Although binary tree has slightly higher storage demand, it can achieve much cheaper probability update, as well as efficient adaptive sampling.

Suppose that there are n data examples. Let π_i be the i-th leaf of $\mathcal{T}_n$. By properly adding "zero" nodes, we can group the nodes at each level in pairs. Figure 1 illustrates an example of such a binary tree with $n=7$. Note that we need to add one more leaf at the bottom level, for grouping it with π_7. According to Algorithm 1, some π_i is changed in each iteration. Hence, all the ancestor nodes of π_i in $\mathcal{T}_n$ should be updated. The update of $\mathcal{T}_n$ can be done in a botton-up way. It is known that the height of $\mathcal{T}_n$ is $\lceil\log n\rceil$. Thus, by updating the tree in a bottom-up approach, the computational cost is $O(\log n)$.

$\mathcal{T}_n$ can contribute to generating a random integer following a given distribution. If we first generate a uniformly distributed random number $r\in[0,1]$, we are supposed to find the index i such that $\sum_{j=1}^{i-1}p_j^k < r < \sum_{j=1}^{i}p_j^k$. By (8) we have

$$\sum_{j=1}^{i}p_j^k = \frac{(1-\alpha_k)i}{n} + \alpha_k\frac{\sum_{j=1}^{i}\pi_j}{\sum_{l=1}^{n}\pi_l},\ \forall i\in[n],$$

where all the partial sums in the right-hand side of the equation are stored in $\mathcal{T}_n$. Thus, we can quickly find the desired i by visiting $\mathcal{T}_n$ in a top-down fashion. Obviously, searching for i also takes $O(\log n)$ time. The specific implementation procedures are provided in the supplementary materials.

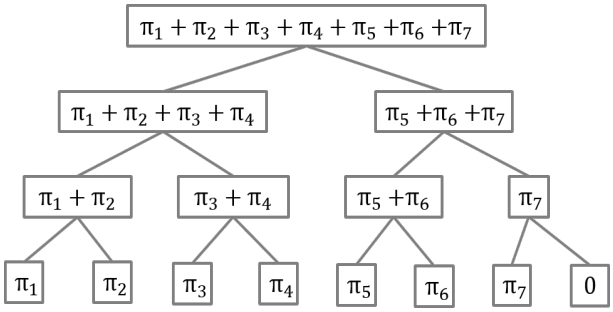

Fig. 1: Example of Binary Tree $\mathcal{T}_7$

To conclude, additional $O(\log n)$ per-iteration cost is needed to implement the adaptive sampling. Since updating the variables already requires $O(d)$ per-iteration cost, our adaptive coordinate sampling method does not increase the order of computational complexity.

III. Convergence Analysis

In this section, we provide the theoretical result that the variance of the stochastic gradient in SGD-AIS is strictly smaller than that in SGD with uniform sampling. To do so, we should first make the following assumptions.

Assumption 1. *Sequence $\{\mathbf{w}_k\}\subseteq\Omega$, where $\Omega\subseteq\mathbb{R}^d$ is a compact set.*

In general, Assumption 1 holds when the objective function is level-bounded. Although stochastic gradient methods are not guaranteed to be descent algorithms, Assumption 1 still holds for all convergent optimization processes which result from bounded sequences. Then, we define an the upper bound of the gradient norm:

$$G := \max_{\mathbf{w}\in\Omega}\left\{\max_{i\in[n]}\{\|\nabla f_i(\mathbf{w})\|_2\}\right\}. \tag{16}$$

Since Ω is compact, we know that G is well-defined, i.e., $G<+\infty$.

Assumption 2. *There exist some constants $\delta>0$ and $\rho>0$ such that*

$$\min_{\mathbf{w}\in\Omega}\left\{\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(\mathbf{w})\|_2\right\}\geq\delta G \tag{17}$$

and

$$\min_{\mathbf{w}\in\Omega}\left\{\frac{1}{2n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\|\nabla f_i(\mathbf{w})\|_2-\|\nabla f_j(\mathbf{w})\|_2\right)^2\right\}\geq\rho G^2. \tag{18}$$

Assumption 2 lower bounds the average gradient norm and the sum of differences of gradient norms. It is straightforward to verify that Assumption 2 implies that $\delta\leq 1$ and $\rho\leq\frac{1}{2}$. The two inequalities in Assumption 2 can be shown to hold with high probability if we assume $f_i(\mathbf{w})=g(\mathbf{x}_i^T\mathbf{w})+\frac{\lambda}{2}\|\mathbf{w}\|_2^2$ where the data $\mathbf{x}_i$ follows the Gaussian or the uniform distribution, and $g(\cdot)$ is a continuous loss function. Both the objective

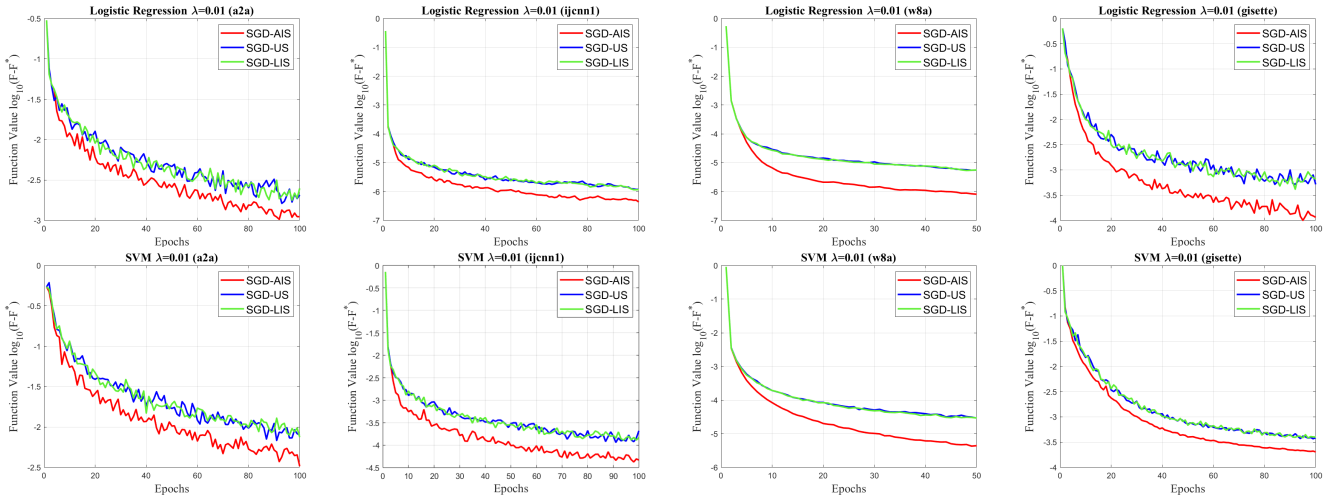

Fig. 2: SGD for Logistic Regression and SVM

functions of logistic regression and support vector machine considered in this paper follows this form.

Assumption 3. *Suppose $N \geq n$, then for all $j \in [n]$ and iteration $k \geq N$, j has been picked in the last N iterations, i.e., $j \in \{i_{k-1}, i_{k-2}, \ldots, i_{k-N}\}$.*

Assumption 3 makes sure that all entries in $\boldsymbol{\pi}$ are updated for at least once within any N consecutive iterations. We show in the supplementary materials that Assumption 3 can hold with high probability for large enough N.

A. Upper Bound on the Variance

We provide the main theorems for our adaptive importance sampling method, whose complete proofs are given in the suplementary materials. We first state the following theorem that plays an key role in establishing the convergence results of SGD-AIS.

Theorem 1. *Suppose that $\mathbf{w}$ is an iterate after N iterations run by Algorithm 1, and $\boldsymbol{p}$ is the most recently updated probability distribution. Under Assumptions 1-3, suppose that the maximum stepsize η in Algorithm 1 satisfies*

$$0 < \eta < \frac{(1-\overline{\alpha})^3 \underline{\alpha} \delta \rho}{(1-\overline{\alpha})^2 \underline{\alpha} NL\rho + \overline{\alpha} NL}. \tag{19}$$

Then, we have

$$\mathrm{Var}_{i\sim \boldsymbol{p}}\left[\frac{1}{np_i}\nabla f_i(\mathbf{w})\right] \leq \mathrm{Var}_{i\sim\mathcal{U}}[\nabla f_i(\mathbf{w})] - \gamma G^2, \tag{20}$$

where $\gamma = \underline{\alpha}\rho - \frac{\overline{\alpha} L\eta N}{(1-\overline{\alpha})^3\delta - (1-\overline{\alpha})^2 L\eta N} \in (0,1)$.

Theorem 1 provides an upper bound on the variance of the stochastic gradient in SGD-AIS. It shows that under proper choice of the stepsize, SGD-AIS achieves a strictly smaller variance than the variance based on uniform sampling, by an amount of at least γG^2. It is important to note that Theorem 1 demonstrates a non-asymptotic result since $\mathbf{w}$ can be an arbitrary iterate. We provide a novel proof to the above theorem by introducing an "auxiliary" variance in the suplementary materials.

B. Convergence Bounds

Equipped with Theorem 1, similar with Theorem 4.7 of [4], we can establish the corresponding convergence bound of SGD-AIS with constant stepsizes for strongly convex problems.

Theorem 2. *Under the Assumption 1-3, suppose that the objective function $F(\mathbf{w})$ is both L-smooth and σ-strongly convex function, and SGD-AIS is run with a fixed stepsize $\eta_k = \eta$ for all $k \in \mathbb{N}$, satisfying*

$$0 < \eta < \frac{(1-\overline{\alpha})^3 \underline{\alpha} \delta \rho}{(1-\overline{\alpha})^2 \underline{\alpha} NL\rho + \overline{\alpha} NL}. \tag{21}$$

Then we have

$$\begin{aligned} &\mathbb{E}[F(\mathbf{w}_k) - F^*] \\ \leq &\frac{\eta L(1-\gamma)G^2}{4\sigma} + (1-2\eta\sigma)^{k-1}\left(F(\mathbf{w}_1) - F^*\right) \\ \xrightarrow{k\to\infty} &\frac{\eta L(1-\gamma)G^2}{4\sigma}. \end{aligned} \tag{22}$$

Theorem 2 shows that the expected function value converge to a neighborhood of the global minimum with linear rate. For vanilla SGD under the same setting, $(1-\gamma)G^2$ in (22) of Theorem 2 is replaced with G^2 [4]. Therefore, the convergence bounds of SGD-AIS are improved by a factor of $1-\gamma < 1$.

Furthermore, if we choose diminishing stepsize, SGD-AIS is guaranteed to converge to the global optimal at the following sublinear rate.

Theorem 3. *Under the same Assumptions 1-3, suppose that SGD-AIS is run with a sequence of diminishing stepsizes such that, for all $k \in \mathbb{N}$, $\eta_k = \beta/(\xi + k)$, where $\beta > 1/(2\sigma)$, $\gamma > 0$ and*

$$\eta_1 < \frac{(1-\overline{\alpha})^3 \underline{\alpha} \delta \rho}{(1-\overline{\alpha})^2 \underline{\alpha} NL\rho + \overline{\alpha} NL}. \tag{23}$$

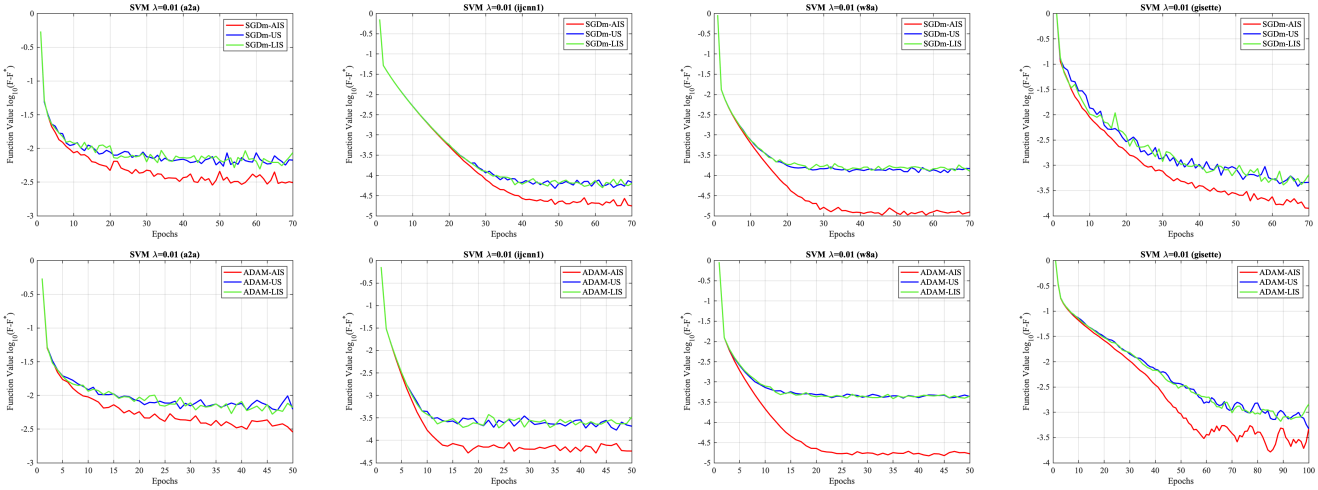

Fig. 3: SGDm and ADAM with Momentum for SVM

Then we have $\mathbb{E}[F(\mathbf{w}_k) - F^*] \leq \nu/(\xi + k)$*, where*

$$\nu := \max\left\{\frac{\beta^2 L(1-\gamma)G^2}{2(2\beta\sigma - 1)}, (\xi+1)(F(\mathbf{w}_1) - F^*)\right\}.$$

Theorem 3 shows that the function value converges to the global minimum with rate $O(1/\epsilon)$. Similarly, comparing with vanilla SGD under the same setting where$(1-\gamma)G^2$ in Theorem 3 is replaced with G^2 [4], the convergence bounds of SGD-AIS are improved by a factor of $1-\gamma < 1$. We also provide more convergence analysis under the nonconvex settings, which are stated and proved in the suplementary materials.

IV. Experiments

In this section, we provide our experimental results to verify the effectiveness of SGD-AIS, SGDm-AIS and ADAM-AIS by applying them to several common learning tasks.

A. Evaluating SGD-AIS

We implement three algorithms, which are SGD-AIS, SGD with uniform sampling (SGD-US), SGD with Lipschitz-based importance sampling (SGD-LIS) [19], [41] using Matlab 2019a. Specifically, for SGD-LIS, we let the fixed probability of sampling the i-th training example be $L_i/\sum_{j=1}^n L_j$, where L_i is the Lipschitz constant of ∇f_i. For SGD-AIS, we set $\underline{\alpha} = 0.3$, $\overline{\alpha} = 0.8$, and $\alpha_k = \underline{\alpha} + (\overline{\alpha} - \underline{\alpha})k/K$, where K is the maximum number of periods we run. For all the three algorithms, we adopt a sequence of best-tuned stepsizes that diminishes with rate $O(1/k)$. See the suplementary materials for more specific values of parameters.

We present the empirical evaluation for two convex learning problems, which are respectively ℓ_2-regularized logistic regression

$$\min_{\mathbf{w}\in\mathbb{R}^d}\left\{\frac{1}{n}\sum_{i=1}^n \log\left(1+\exp\left(-y_i\mathbf{x}_i^\top\mathbf{w}\right)\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2\right\} \quad (24)$$

and support vector machine (SVM) based on squared hinge loss

$$\min_{\mathbf{w}\in\mathbb{R}^d}\left\{\frac{1}{n}\sum_{i=1}^n \left(\left[1-y_i\mathbf{x}_i^\top\mathbf{w}\right]_+\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2\right\}. \quad (25)$$

Both of these two objective functions are strongly convex and have Lipschitz continuous gradients. The experiments are performed on the datasets a2a, ijcnn1, w8a, and gisette [6], whose sizes are given in the supplementary materials.

Figure 2 illustrates the performance of the three algorithms applied to convex learning problems, i.e., logistic regression and support vector machine. We consider $\log_{10}(F - F^*)$ for the first 100 epochs (i.e., $100n$ iterations), where F^* denotes the optimal function value. We take the average function values by running each algorithm 30 times with the same initial point. As it can be seen, SGD-US and SGD-LIS have roughly the same sublinear rate of convergence, since the used global Lipschitz constants in Lipschitz-based sampling cannot capture much local information, and even the L_i's do not differ from each other a lot. At the initial stage, SGD-AIS adopts the sampling distribution that is close to the uniform distribution, thus it has similar convergence rates with the other two algorithms. SGD-AIS soon achieves distinctively higher convergence accuracy after several epochs, since the sampling distribution $\boldsymbol{p}$ gets closer to the sub-optimal distribution $\mathcal{G}$. In general, SGD-AIS has apparently better performance for all the chosen datasets, whose n and d vary from $n \gg d$ to $n \approx d$. One possible reason is that, under these cases, Assumption 2 will hold with larger δ and ρ, which leads to larger γ in Theorem 1 and hence better convergence.

For fair comparisons of the empirical results, we also provide the runtime of the experiments. All our experiments are conducted based on an Intel i5 processor with 3.1GHz main frequency. Table I and II shows the specific runtime (in seconds) per epoch. As expected, SGD-LIS and SGD-AIS takes slightly more computational time than SGD-US, while in general they are very close to each other. Thus, the

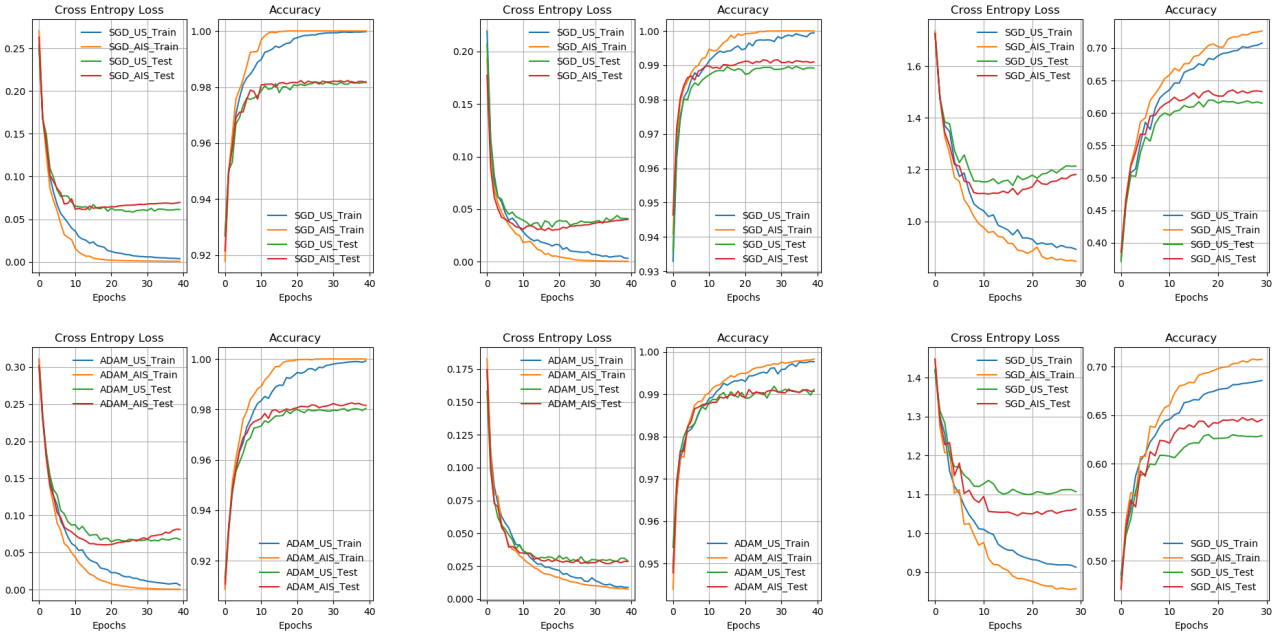

Fig. 4: SGDm and ADAM for Neural Networks (Column 1-3: MLP (MNIST), LeNet-5 (MNIST), CNN (Cifar-10); Row 1: SGD-US v.s. SGD-AIS; Row 2: ADAM-US v.s. ADAM-AIS)

sharper convergence performance of adaptive sampling will not be apparently counteracted.

B. Evaluating SGDm-AIS and ADAM-AIS

For SGDm-AIS and ADAM-AIS, we evaluate their performance on both convex and nonconvex learning problems.

1) Experiments on Convex Learning: We conduct experiments on SVM with squared hinge loss (25) using Matlab. We adopt constant stepsizes, and set $\underline{\alpha}$, $\overline{\alpha}$, α_k to be the same values as those in the section 6.1, and the weight θ in (10) is set as 0.9. Figure 3 presents the convergence performance of SGDm, ADAM, and their importance sampling derivatives. We can observe that for both SGDm and ADAM, our adaptive sampling strategy still exhibits significantly sharper rates of convergence than the algorithms with uniform sampling or non-uniform stationary sampling.

2) Experiments on Neural Networks: In order to evaluate the performance of SGDm-AIS and ADAM-AIS in more complex nonconvex problem, we further conduct simulation on several classical neural networks, including MLP (multilayer perceptron), CNN (convolutional neural network), and LeNet-5 [18]. We use two common benchmark datasets, namely MNIST and CIFAR-10 for our experiments.

We implement SGDm-AIS, SGDm-US (uniform sampling), ADAM-AIS and ADAM-US (uniform sampling) in PyTorch. MNIST is applied to a three-layer MLP, which is a fully connected neural network with ReLU activition (i.e., the number of hidden neurons is 784-500-256-10), and the LeNet-5. Besides, CIFAR-10 is applied to a CNN, whose structural details are given the suplementary materials. The batch sizes we choose are 8 for MNIST and 16 for CIFAR-10. Moreover,

TABLE I: Runtime (s) for Logistic Regression

	a2a	ijcnn1	w8a	gisette
SGD-US	0.0031	0.049	0.212	0.357
SGD-LIS	0.0033	0.062	0.225	0.360
SGD-AIS	0.0035	0.064	0.231	0.369

TABLE II: Runtime (s) for SVM

	a2a	ijcnn1	w8a	gisette
SGD-US	0.0027	0.0454	0.184	0.311
SGD-LIS	0.0030	0.0575	0.195	0.314
SGD-AIS	0.0032	0.0609	0.204	0.328

we adjust the learning rate during training by reducing the learning rate according to a pre-defined schedule. Here, we choose the step decay schedule, which drops the learning rate by a factor $\rho < 1$ every few epochs. More specific choices of parameters are also provided in the suplementary materials.

It is observed from Figure 4 that SGDm-AIS and ADAM-AIS achieves evidently lower training error and higher training accuracy compared with SGDm-US and ADAM-US respectively, for all the three neural nwtworks. In terms of test error and test accuracy, SGDm-AIS and ADAM-AIS exhibits a more stable pattern than SGDm-US ans ADAM-US.

V. Conclusion

In this paper, we have investigated an importance sampling strategy with three notable attributes, i.e., Lipschitz-independent, computationally efficient, and adaptive to the

local function property. We provide a non-asymptotic upper bound for the variance of the stochastic gradients, and verify that the reduced variance leads to faster convergence rates both in theory and practice. We also extend the adaptive importance sampling method to SGD with momentum and ADAM. We perform extensive experiments on SGD-AIS, SGDm-AIS and ADAM-AIS under convex and nonconvex settings. The empirical results exhibit obviously superior convergence performance than traditional algorithms based on uniform and non-uniform stationary sampling. We think our adaptive sampling strategy has much potential to accelerate training process of learning tasks with larger scale.

REFERENCES

[1] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research*, 18(1):8194–8244, 2017.

[2] Wangpeng An, Haoqian Wang, Qingyun Sun, Jun Xu, Qionghai Dai, and Lei Zhang. A pid controller approach for stochastic optimization of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8522–8531, 2018.

[3] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[4] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[5] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

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

[7] Dominik Csiba, Zheng Qu, and Peter Richtárik. Stochastic dual coordinate ascent with adaptive probabilities. In *International Conference on Machine Learning*, pages 674–683, 2015.

[8] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[9] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4943–4953, 2019.

[10] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[11] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 689–699, 2018.

[12] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[13] Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5200–5209, 2019.

[14] Samuel Horváth and Peter Richtarik. Nonconvex variance reduced optimization with arbitrary sampling. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2781–2789, 2019.

[15] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[16] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of 31st International Conference on Learning Representations*, 2014.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in Neural Information Processing Systems*, pages 1017–1025. 2014.

[20] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[21] Yurii Nesterov. *Introductory lectures on convex programming: A Basic course*. Springer US, 2004.

[22] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[23] Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[24] Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2613–2621, 2017.

[25] Guillaume Papa, Pascal Bianchi, and Stéphan Clémençon. Adaptive sampling for incremental optimization using stochastic gradient descent. In *International Conference on Algorithmic Learning Theory*, pages 317–331. Springer, 2015.

[26] Dmytro Perekrestenko, Volkan Cevher, and Martin Jaggi. Faster coordinate descent via adaptive importance sampling. In *Artificial Intelligence and Statistics*, pages 869–877, 2017.

[27] Xun Qian, Zheng Qu, and Peter Richtárik. SAGA with arbitrary sampling. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5190–5199, 2019.

[28] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[29] Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484, 2016.

[30] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407, 1951.

[31] Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

[32] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[33] Fanhua Shang, Kaiwen Zhou, Hongying Liu, James Cheng, Ivor Tsang, Lijun Zhang, Dacheng Tao, and Jiao Licheng. Vr-sgd: A simple stochastic variance reduction method for machine learning. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[34] Zebang Shen, Hui Qian, Tengfei Zhou, and Tongzhou Mu. Adaptive variance reducing for stochastic gradient descent. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 1990–1996, 2016.

[35] Anthony Man-Cho So and Zirui Zhou. Non-asymptotic convergence analysis of inexact gradient methods for machine learning without strong convexity. *Optimization Methods and Software*, 32(4):963–992, 2017.

[36] Sebastian U Stich, Anant Raj, and Martin Jaggi. Safe adaptive importance sampling. In *Advances in Neural Information Processing Systems*, pages 4381–4391, 2017.

[37] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[38] Paul Tseng. An incremental gradient (-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

[39] Bingzhe Wu, Shiwan Zhao, Guangyu Sun, Xiaolu Zhang, Zhong Su, Caihong Zeng, and Zhihong Liu. P3sgd: Patient privacy preserving sgd for regularizing deep cnns in pathological image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2099–2108, 2019.

[40] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[41] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *Proceedings of 22nd International Conference on Machine Learning*, pages 1–9. 2015.