Stable Gradient Descent

Yingxue Zhou **Sheng Chen** **Arindam Banerjee**

{zhou0877@umn.edu, shengc, banerjee@cs.umn.edu}

Department of Computer Science & Engineering
University of Minnesota, Twin Cities

Abstract

The goal of many machine learning tasks is to learn a model that has small population risk. While mini-batch stochastic gradient descent (SGD) and variants are popular approaches for achieving this goal, it is hard to prescribe a clear stopping criterion and to establish high probability convergence bounds to the population risk. In this paper, we introduce ***St**able **G**radient **D**escent* which validates stochastic gradient computations by splitting data into training and validation sets and reuses samples using a differential private mechanism. StGD comes with a natural upper bound on the number of iterations and has high-probability convergence to the population risk. Experimental results illustrate that StGD is empirically competitive and often better than SGD and GD.

1 INTRODUCTION

The primary goal in several machine learning tasks is to learn a model with finite training samples that generalizes well to unseen instances. One typically attempt to solve the following optimization problem which finds a minimizer $w^\star$ of the *population risk* F over some model class $\mathcal{W}$:

$$w^\star \in \operatorname*{argmin}_{w\in\mathcal{W}} \; F(w) \triangleq \mathbb{E}_{z\sim\mathcal{P}}[l(w,z)] \;, \qquad (1)$$

where $z \in \mathcal{Z}$ is a data point in domain $\mathcal{Z}$ following the unknown distribution $\mathcal{P}$, and $l : \mathcal{W}\times\mathcal{Z} \mapsto \mathbb{R}$ is a certain loss function associated with the learning problem. For example, in classification problems $z = (\mathbf{x}, y)$ is an instance-label pair, w denotes a classifier, and $l(w, z)$ can be the hinge loss or logistic loss.

Due to the unavailability of distribution $\mathcal{P}$, the challenge of a learning algorithm is to search for an approximate minimizer $\hat{w}_n$ of the risk F based *only* on a set of finite samples $Z_n = \{z_1, z_2, ..., z_n\}$. A good criterion for quantifying the quality of $\hat{w}_n$ is through the *excess risk*:

$$F(\hat{w}_n) - F(w^\star) \;. \qquad (2)$$

A learning algorithm for obtaining $\hat{w}_n$ should have small excess risk. Being a function of the random set of samples Z_n, $F(\hat{w}_n)$ is a random variable and a good learning algorithm is expected to have small excess risk with high probability.

In the literature, there are several approaches to tackle the problem. The classical approach is *empirical risk minimization* (ERM) (Shalev-Shwartz and Ben-David [2014]) which aims to find an empirical minimizer defined as $\hat{w}_n^{ERM} \in \operatorname{argmin}_{w\in\mathcal{W}} \; \hat{F}(w) \triangleq \frac{1}{n}\sum_{j=1}^{n} l(w, z_j)$. Usually first-order iterative optimization methods such as *gradient descent* (GD) are used to obtain the minimizer. However, GD uses all samples to compute gradients in each iteration and is quite slow for large datasets. A popular approach in modern machine learning is *stochastic gradient descent* (SGD)(Zhang [2004];Rakhlin et al. [2012];Hazan and Kale [2014]). SGD

minimizes the population risk directly by descending along stochastic gradients, computed based on a single sample or a mini-batch of samples. The stochastic gradient equals the population gradient in expectation (Shalev-Shwartz and Ben-David [2014]). Convergence of SGD is typically analyzed in expectation rather than with high probability with a few notable exceptions(Rakhlin et al. [2012]). In practice, since the improvements in the objective function value is non-monotonic, it is hard to prescribe a theoretically well grounded stopping criterion.

Conceptually, a mini-batch SGD algorithm would be much more well behaved if we had access to n fresh samples in each iteration. In such a setting, one would be able establish high probability bounds on the sample gradients staying close to the population gradient across all iterations, leading to high probability bounds on the excess risk. In this paper, we introduce *Stable Gradient Descent* (StGD), which behaves similar to the ideal case of n fresh samples per iteration using ideas from *adaptive data analysis* (Dwork et al. [2015c]) and *differential privacy* (Dwork and Roth [2014]). Each iteration leads to a small privacy loss which, unlike SGD, automatically puts a bound on the number of iterations StGD can be run. We present basic and mini-batch StGD and provide high probability bounds on the excess risk for different types of convex loss functions.

The main idea in StGD is simple: separate the available samples into a training and a validation set, compute stochastic gradient on both, and check if they are close. If they are indeed close, there is confidence in that descent direction and StGD proceeds with the descent. On the other hand, if they are not close, there is lack of confidence in the descent direction, and StGD uses a noisy version of the estimated gradient to do descent. The challenge in naively carrying out the simple idea is that the algorithm may overfit on both the training and the validation set. As a result, StGD carries out the comparison of training and validation gradients using a differentially private mechanism, which allows StGD to reuse the same samples over iterations but still get high probability bounds across all iterations as if the samples were fresh.

The remainder of the paper is organized as follows. Section 2 describes related work. We present basic StGD in Section 3, and present min-batch StGD in Section 4, along with high probability excess risk bounds. We present experimental results in Section 5, and conclude in Section 6. All technical proofs are deferred to the supplementary material.

2 RELATED WORK

ERM and SGD: For ERM, a common algorithm is gradient descent (Shalev-Shwartz and Ben-David [2014]) which computes the full gradient of the empirical risk and takes a step along it at each iteration. The performance of ERM is usually measured in terms of the uniform convergence of $\hat{F}(w)$ to $F(w)$ over $\mathcal{W}$. (Hardt et al. [2015]) analyzed the stochastic gradient method (SGM) for ERM in terms of stability and optimization error: $\mathbb{E}[F(\hat{w}_n)] - F(w^\star) \leq \epsilon_{opt}(w) + \epsilon_{stab}$. They demonstrated that for L-Lipschitz continuous function, SGM has the convergence rate of $O(L/\sqrt{n})$. SGD minimizes the population risk by allowing the optimization procedure to take a step along a random direction, as long as the expected value of the direction is the population gradient (Shalev-Shwartz and Ben-David [2014]). In this case, with n to be the number of stochastic gradient computations. For strongly convex functions, SGD can achieve an expectation risk bound of rate $O(1/n)$ (Hazan and Kale [2014],Rakhlin et al. [2012]). (Rakhlin et al. [2012]) also presented a similar high probability bound for SGD.

Differential Privacy: Informally, differential private analysis ensures that the outcome of analysis on two nearly identical input datasets (different on a single component) should also be nearly identical. As a result, an analyst will not be able to distinguish any single data by comparing the change of output. In the context of machine learning, this randomized algorithm $\mathcal{M}$ can be a learning algorithm that outputs a classifier $\mathcal{M}(D) = f$ where D is the training set. Some work (Chaudhuri et al. [2011], Bass-

ily et al. [2014]) introduced differential privacy to ERM to protect sensitive information about training data. (Dwork et al. [2015b], Dwork et al. [2015a]) introduced differential privacy to adaptive data analysis (ADA). In ADA, analyst test adaptively generated hypotheses on one holdout set where those hypotheses have dependence on the holdout set. To ensure the repeatedly used holdout set to provide valid validations, they designed a *Thresholdout* mechanism which allows the analyst to query the holdout set via a differentially private way. They showed differentially private reused holdout set maintains the statistical nature of fresh sample.

The main contribution of this paper is, we introduce differential privacy to gradient descent by applying *Thresholdout* to the training set. We show that the training set can be reused and maintains the statistical nature of fresh sample in all iterations. Mathematically speaking, gradients computed on the differentially-private reused training set concentrate around the population value with high probability. We exploit the concentration property to derive high-probability risk bounds of StGD.

3 STABLE GRADIENT DESCENT

3.1 PRELIMINARIES

We consider the problem of minimizing the population risk defined in Equation 1. We denote $\nabla l(w, z)$ as the gradient of $l(w, z)$. We use $\nabla_i l(\cdot)$ to denote the i-th coordinate of $\nabla l(\cdot)$. Besides, we use $g_i(w)$, $\tilde{g}_i(w)$ and $\hat{g}_i(w)$ to be the i-th coordinate of $g(w)$ (population gradient), $\tilde{g}(w)$ (gradient computed by StGD) and $\hat{g}(w)$ (empirical gradient), respectively, for $i \in \{1, ..., d\}$, where d is the dimension of w. For example, given the sample set S, $\hat{g}(w) = \sum_{z_j \in S} \nabla l(w, z_j)/|S|$. We consider some special cases of $F(w)$ with the following assumptions(Boyd and Vandenberghe [2004]):

1. Convex and L-Lipschitz: Function F is convex with L-Lipschitz if for all $w, w\prime \in \mathcal{W}$ and $L \geq 0$:

$$|F(w\prime) - F(w)| \leq L\|w\prime - w\|.$$

2. Strongly Convex: Formally, a function F is α-strongly convex, if for all $w, w\prime \in \mathcal{W}$ and any subgradient $g(w)$ of F at w, we have

$$F(w\prime) \geq F(w) + (w\prime - w)^T g(w) + \frac{\alpha}{2}\|w\prime - w\|^2.$$

3. Smooth: We say a function F is β-smooth, if $w, w\prime \in \mathcal{W}$ and any subgradient $g(w)$ of F at w, we have

$$F(w\prime) \leq F(w) + (w\prime - w)^T g(w) + \frac{\beta}{2}\|w\prime - w\|^2.$$

Algorithm 1 Stable Gradient Descent (StGD) Algorithm

1: **Input**: Dataset S, certain loss $l(\cdot)$, initial point w_0.
2: **Set**: Noise variance σ, iteration time T, step size η.
3: Separate S randomly and evenly into S_t and S_h.
4: **for** $s = 0, ..., T$ **do**
5: Run **DPGC**(S_t, S_h, w_s, σ, $l(\cdot)$) to compute gradient $\tilde{g}(w_s)$.
6: $w_{s+1} = w_s - \eta\tilde{g}(w_s)$.
7: **end for**

3.2 STGD ALGORITHM

We present StGD in two parts: Algorithm 1 and Algorithm 2 (**DPGC**). To simplify, we suppose there are $2n$ available samples $S \sim \mathcal{P}^{2n}$. The StGD randomly and evenly separates them into two datasets: training set S_t and validation set S_h, both of which are of size n. We set a noise parameter σ and the total iterations T. We analyze the optimal values of parameters σ and T in the next section. Starting from initial point w_0, at each s-th iteration, StGD runs **DPGC** (Algorithm 2) to query the training set S_t in order to obtain an estimated gradient $\tilde{g}(w_s)$, then updates the w_{s+1} based on $\tilde{g}(w_s)$ (line 5, 6 in Algorithm 1).

We present the pseudo-code of **DPGC** in Algorithm 2. **DPGC** unrestrictedly accesses the validation set S_h, but accesses S_t via a differentially private way: Given w_s, **DPGC** first computes gradients on S_t and S_h: $g^t(w_s) = \sum_{z_i \in s_t} \nabla l(w_s, z_i)/|S_t|$, $g^h(w_s) =$

Algorithm 2 Differentially Private Gradient Computation (DPGC)

1: **Input**: Dataset S_t and S_h, parameter w_s, noise variance σ, loss $l(\cdot)$.
2: Compute gradients $g^t(w_s)$ and $g^h(w_s)$: $g^t(w_s) = \sum_{z_i \in s_t} \nabla l(w_s, z_i)/|S_t|$, $g^h(w_s) = \sum_{z_i \in s_h} \nabla l(w_s, z_i)/|S_h|$.
3: **for** i= 1,...,d **do**
4: Sample $\xi \sim Lap(\sigma)$, $\gamma \sim Lap(2 \cdot \sigma)$, $\tau \sim Lap(4 \cdot \sigma)$.
5: **if** $|g_i^t(w_s) - g_i^h(w_s)| > \gamma + \tau$ **then**
6: $\tilde{g}_i(w_s) = g_i^t(w_s) + \xi$.
7: **else**
8: $\tilde{g}_i(w_s) = g_i^h(w_s)$.
9: **end if**
10: **end for**
11: **Return**: $\tilde{g}(w_s)$.

$\sum_{z_i \in s_h} \nabla l(w_s, z_i)/|S_h|$. Second, for each coordinate $i \in \{1, ..., d\}$, **DPGC** validates $g_i^t(w_s)$ with $g_i^h(w_s)$ (line 5-line 8 in Algorithm 2): If their absolute difference is beyond the threshold $\gamma + \tau$, **DPGC** outputs $g_i^t(w_s)$ with noise. Otherwise, **DPGC** returns $g_i^h(w_s)$.

3.3 CONVERGENCE ANALYSIS

We assume that for every i-th coordinate, the gradient function $|\nabla_i l(w, z)| \leq G$ for a fixed constant G. Given an n-sample set $S \in \mathcal{Z}^n$ and a fixed w_0 that is chosen independent of the dataset S, $\hat{g}_i(w_0) = \sum_{z_j \in S} \nabla_i l(w_0, z_j)/n$ and $g_i(w_0) = \mathbb{E}_{z \sim \mathcal{P}}[\nabla_i l(w_0, z_j)]$, by Hoeffding's bound, we have the concentration as

$$P\{|\hat{g}_i(w_0) - g_i(w_0)| \geqslant \sigma\} \leqslant 2\exp\left(\frac{-2n\sigma^2}{4G^2}\right). \quad (3)$$

In general, updating w_1 through typical gradient descent: $w_1 = w_0 - \eta\hat{g}(w_0)$, the above concentration bound does not hold for $\hat{g}_i(w_1) = \sum_{z_j \in S} \nabla_i l(w_1, z_j)/n$, because w_1 is no longer independent of dataset S. In the next lemma, we demonstrate that $w_1, w_2, ..., w_T$ updated by a differential private mechanism have similar concentration bounds as described in Equation 3.

Lemma 1. *Let $\mathcal{M}$ be an ϵ-differentially private gradient descent algorithm and $S_t \sim \mathcal{P}^n$ be the training set. Let $w_s = \mathcal{M}(S_t)$ be the corresponding output for $s \in 1, ..., T$ and $\hat{g}(w_s)$ be the empirical gradient on S_t. For any $\sigma > 0$, $i \in 1, ..., d$ and $s \in 1, ..., T$, setting $\epsilon \leqslant \frac{\sigma}{2G}$ ensures*

$$P\{|\hat{g}_i(w_s) - g_i(w_s)| \geqslant \sigma\} \leqslant 6\sqrt{2}\exp\left(\frac{-n\sigma^2}{4G^2}\right). \quad (4)$$

Lemma 1 illustrates that differential privacy enables the reused training set to maintain the statistical guarantee as a fresh set under the condition that the privacy parameter ϵ is bounded by the estimation error σ. Next, we analyze the privacy parameter ϵ of StGD.

Lemma 2. *StGD satisfies $\frac{2TG}{n\sigma}$-differentially private.*

In order to achieve the gradient concentration bound described in Lemma 1 by considering the guarantee of Lemma 2 (i.e. to guarantee that for every w_s, we have $P\{|\hat{g}_i(w_s) - g_i(w_s)| \geqslant \sigma\} \leqslant 6\sqrt{2}\exp(\frac{-n\sigma^2}{4G^2})$), we need to set $\frac{2TG}{n\sigma} \leqslant \frac{\sigma}{2G}$ so that we achieve ϵ-differential privacy for $\epsilon \leqslant \frac{\sigma}{2G}$. As a result, we get the upper bound of iteration time T in StGD as $T = \frac{\sigma^2 n}{4G^2}$. Next theorem shows that across all iterations, gradients produced by StGD maintain high probability concentration bounds.

Theorem 1. *Given parameter $\sigma > 0$, let $w_1, w_2, ..., w_T$ be the adaptively updated points by StGD and $\tilde{g}(w_1), ..., \tilde{g}(w_T)$ be the corresponding output gradient. If we set $T = \frac{\sigma^2 n}{4G^2}$, then for all $s \in 1, ..., T$ and for all $t > 0$, we have*

$$\begin{aligned} P\{ \quad & \|\tilde{g}(w_s) - g(w_s)\|^2 \geqslant d(6t+1)^2\sigma^2\} \\ & \leqslant 2d\exp(-t) + 6\sqrt{2}d\exp\left(\tfrac{-n\sigma^2}{4G^2}\right). \end{aligned} \quad (5)$$

Theorem 1 concludes that the gradient $\tilde{g}(w_s)$ produced by StGD concentrates to the population gradient $g(w_s)$ and the concentration error is tightly around $(6t+1)^2\sigma^2$. Increasing noise parameter σ increases the privacy guarantee as well as the total number of iterations, but also increases the concentration error. Decreasing σ has the opposite effect. We consider StGD in two cases: 1) F is L-Lipschitz and α-strongly convex; 2) F is β-smooth and α-strongly convex. For these two cases, we present the best value of σ for the trade-off between statistical rate

and optimization rate which depends on number of iterations in order to achieve the optimal risk bound. To simplify the result, we use the notation $\rho_{n,d} = \ln n + \ln d$.

Theorem 2. *For L-Lipschitz and α-strongly convex function F, given $2n$ available samples, set noise parameter $\sigma^2 = 4G^2\rho_{n,d}/\sqrt{n}$, step size $\eta_s = \frac{2}{\alpha(s+1)}$ and iteration time $T = \rho_{n,d}\sqrt{n}$ for StGD. Let $\hat{w}_n = \sum_{s=0}^{T} w_s/(T+1)$, StGD achieves:*

$$F(\hat{w}_n) - F(w^\star) \leqslant O\left(\frac{\ln(\sqrt{n}\rho_{n,d})}{\sqrt{n}\rho_{n,d}}\right) + O\left(\frac{d\rho_{n,d}^3}{\sqrt{n}}\right), \tag{6}$$

with probability at least $1 - O(\frac{\rho_{n,d}}{\sqrt{n}})$

The first term of the risk bound in Theorem 2 corresponds to typical strongly convex optimization rate $O(\ln T/T)$ (Bubeck [2015]) ($T = \rho_{n,d}\sqrt{n}$ in our case) and is similar to the high probability bound of SGD analyzed in Rakhlin et al. [2012]. The second term comes from the statistical error that depends on available sample size n and dimension d.

Theorem 3. *For β-smooth and α-strongly convex function F, given $2n$ available samples, set noise parameter $\sigma^2 = \frac{\rho_{n,d}(4G^2\alpha+\beta)^2}{n\alpha\beta}$, step size $\eta = \frac{1}{\alpha+\beta}$ and iteration time $T = (\kappa + \frac{1}{\kappa} + 2)\rho_{n,d}$ where $\kappa = \beta/\alpha$. Let $\hat{w}_n = w_T$ be the output of StGD, we have the following excess risk bound:*

$$F(\hat{w}_n) - F(w^\star) \leqslant O\left(\frac{\|w_1 - w^\star\|^2}{n}\right) + O\left(\frac{d\rho_{n,d}^3}{n}\right) \tag{7}$$

with probability at least $1 - O\left(\frac{\rho_{n,d}}{n^4 d^3}\right)$.

The risk bound in Theorem 3 is also composed of optimization term and statistical term (same for the subsequent theorems). Factor $\|w_1 - w^\star\|^2$ implies a good initial point can be beneficial. In terms of computational complexity, StGD repeats $O(\ln n)$ iterations on n samples. It requires a complexity of $O(n \ln n)$ gradient computations.

4 MINI-BATCH EXTENSIONS

In this section, we extend StGD to its mini-batch version for large-scale machine learning tasks. We first introduce a simple *mini-batch SGD* algorithm, then we present the differentially private algorithm *mini-batch StGD*.

Algorithm 3 mini-batch SGD

1: **Input**: Dataset S, loss $l(\cdot)$, initial point w_0
2: **Set**: Step size η, batch size m.
3: Separate S into T parts: $S_0, ..., S_{T-1}$ with m samples each part.
4: **for** $s = 0, ..., T-1$ **do**
5: Compute gradient $\hat{g}(w_s)$ on S_s
6: $w_{s+1} = w_s - \eta_s \cdot \hat{g}(w_s)$
7: **end for**

The mini-batch SGD algorithm is described in Algorithm 3. The available set is first partitioned into T batches with m samples each batch. Then Algorithm 3 updates w_s based on the gradient computed on each batch. Mini-batch SGD terminates after a single pass over all batches. The following theorem analyzes the risk bound of mini-batch SGD.

Theorem 4. *Given $2n$ available samples, mini-batch SGD can achieve the following:*

*1. F is L-**Lipschitz and** α-**strongly convex**: If we set the step size $\eta_s = \frac{2}{\alpha(s+1)}$, batch size $m = \sqrt{n}$ and iteration time $T = 2n/m$, output $\hat{w}_n = \sum_{s=1}^{T} w_s/T$ of mini-batch SGD satisfies:*

$$F(\hat{w}_n) - F(w^\star) \leqslant O\left(\frac{\ln(\sqrt{n}+1)}{\sqrt{n}}\right) + O\left(\frac{d\ln\sqrt{n}}{\sqrt{n}}\right) \tag{8}$$

with probability at least $1 - d/\sqrt{n}$.

*2. F is β **smooth and** α-**strongly convex**: If we set the step size $\eta = \frac{1}{\alpha+\beta}$, $m = \frac{\alpha\beta n}{(\alpha+\beta)^2 \ln n}$, $T = 2n/m$, output $\hat{w}_n = w_T$ of StGD satisfies:*

$$F(\hat{w}_n) - F(w^\star) \leqslant O\left(\frac{\|w_1 - w^\star\|^2}{n}\right) + O\left(\frac{d\ln^2 n}{n}\right) \tag{9}$$

with probability $1 - O\left(\frac{\ln n}{n}\right)$.

(Frostig et al. [2015]) proposed a variant of mini-batch SGD that also does a single pass of the available data. For smooth and strongly convex function, they established a convergence rate of $O(1/n)$ in expectation that is similar to the rate in Theorem 4.

Algorithm 4 mini-batch StGD

1: **Input**: Dataset S, loss $l(\cdot)$, initial point w_0
2: **Set**: Step size η, batch size m, inner iterations T_1, noise σ
3: Separate S into T parts: $S_0, ..., S_{T-1}$ with m samples each part.
4: **for** $s = 0, ..., T-1$ **do**
5: $\quad w_{s+1} = \mathbf{StGD}(w_s, S_s, \eta, T_1, \sigma)$
6: **end for**

The mini-batch version of StGD is given in Algorithm 4 (mini-batch StGD) that is also a private version of mini-batch SGD: For each batch S_s, where $s \in \{0, ..., T-1\}$, call StGD to query S_s and update w_{s+1} as the initial point for next batch S_{s+1}. In each call, there are T_1 inner iterations that StGD queries S_s for T_1 times through DPGD. Let $\tilde{w}_0 = w_s$ as the initial point in each call, then sub-algorithm StGD updates $\tilde{w}_{k+1} = \tilde{w}_k + \eta\tilde{g}(\tilde{w}_k)$ for $k = \{0, ..., T_1 - 1\}$ and $w_{s+1} = \tilde{w}_{T_1}$.

Theorem 5. *Given $2n$ available samples, mini-batch StGD can achieve the following:*

*1. F is L-**Lipschitz and** α-**strongly convex**: If we set the step size $\eta_s = \frac{2}{\alpha(s+1)}$, batch size $m = \sqrt{n}$, $T = 2n/m$, noise parameter $\sigma^2 = 8G^2 \ln n/\sqrt{n}$ and $T_1 = \ln n$, output $\hat{w}_n = \sum_{s=1}^{T} w_s/T$ of mini-batch StGD satisfies:*

$$F(\hat{w}_n) - F(w^\star) \leqslant O\left(\frac{\ln(\sqrt{n}+1)}{\sqrt{n}\ln n}\right) + O\left(\frac{\ln^3 n}{\sqrt{n}}\right) \tag{10}$$

with probability at least $1 - d/\sqrt{n}$.

*2. F is β **smooth and** α-**strongly convex**: If we set the step size $\eta = \frac{1}{\alpha+\beta}$, $m = \frac{\alpha\beta n}{(\alpha+\beta)^2 \ln n}$, $T = 2n/m$, $T_1 = \ln n$, noise parameter $\sigma^2 = \frac{4G^2(\alpha+\beta)^2(\ln n)^2}{\alpha\beta n}$, output $\hat{w}_n = w_T$ of mini-batch StGD satisfies:*

$$F(\hat{w}_n) - F(w^\star) \leqslant O\left(\frac{\|w_1 - w^\star\|^2}{n^{\ln n}}\right) + O\left(\frac{d\ln^4 n}{n}\right) \tag{11}$$

with probability at least $1 - O\left(\frac{\ln^2 n}{n}\right)$.

Theorem 5 shows that, compared to the basic mini-batch SGD (Theorem 4), the private version improves the rate of the first term for both types of functions.

5 EXPERIMENTS

In this section, we conduct experiments to evaluate performances of the proposed algorithms on artificial data and real world data. We divide our experiments into three sets to address questions: (i) How do the StGD and the mini-batch StGD perform regarding the convergence to the population optimum? (ii) For small datasets, how does StGD perform compared to SGD and GD. (iii) For large datasets, does mini-batch StGD outperform SGD and mini-batch SGD? After discussing the experimental setup, we evaluate these questions empirically in Sections 5.2, 5.3, and 5.4 respectively.

5.1 EXPERIMENTAL SETTING

Datasets: We use both artificial datasets and real-world datasets for our experiments. We discuss the datasets in two categories: the small datasets (i.e., *small artificial dataset, breast cancer, diabetes and german.numer*) for StGD, SGD and GD, and large datasets (i.e., *large artificial dataset, cove type, rcv1 and real-sim*) for mini-batch StGD, SGD and mini-batch SGD. All the real-world datasets are from LIBSVM (Chang and Lin [2011]). The real-world datasets are described in the Table 1. The small artificial dataset, consists of 50 features and one label: $z_i = (\mathbf{x}_i, y_i) \in \mathbb{R}^{50} \times \{1, -1\}$. The large artificial dataset consists of 500 features and one label: $z_i = (\mathbf{x}_i, y_i) \in \mathbb{R}^{500} \times \{1, -1\}$. The value of each feature is random noise, drawn i.i.d. from normal distribution $N(0, 1)$. To generate the label, we first set an optimal minimizer $w^\star \in \mathbb{R}^{50}$ for small datasets and $w^\star \in \mathbb{R}^{500}$ for large datasets. Then, we draw the label y_i corresponding to $\mathbf{x}_i$ from the Bernoulli distribution $y_i \sim B(\frac{1}{1+exp(-w^{\star T}\mathbf{x}_i)}, \frac{exp(-w^{\star T}\mathbf{x}_i)}{1+exp(-w^{\star T}\mathbf{x}_i)})$.

Evaluation Metrics: We measure the performance of these algorithms for binary classification problem with linear models. We focus on the smooth and strongly convex loss function case and define the loss function F to be the

Table 1: Datasets

Datasets	Data size	Features
breast cancer	683	10
diabetes	768	8
german.numer	1000	24
cove type	581012	54
rcv1	697641	47236
real-sim	72309	20958

logistic loss. Thus, the population risk is

$$F(w) = \mathbb{E}[\ln(1 + \exp(-y_i w^T \mathbf{x}_i))]. \quad (12)$$

Given a training set of instance-label pairs $\{\mathbf{x}_i, y_i\}_{i=1}^n$ the empirical risk is

$$\hat{F}(w) = \frac{1}{n}\sum_{i=1}^{n} \ln(1 + \exp(-y_i w^T \mathbf{x}_i))]. \quad (13)$$

The population optimum is

$$F(w^\star) = \mathbb{E}[\ln(1 + \exp(-y_i w^{\star T} \mathbf{x}_i))]. \quad (14)$$

We use $\hat{w}_n$ to be the output trained on n samples. We evaluate these algorithms in terms of the excess risk $F(\hat{w}_n) - F(w^\star)$ and test error rate for artificial datasets. Since we cannot know the population optimum $F(w^\star)$ out of the expectation, we let the loss computed on a large fresh set (2000 fresh samples for small data case, 20000 fresh samples for large data case) to represent the population risk. As for the real-world datasets, since the population minimizer is unknown, we evaluate these algorithms based on the test loss and test error rate.

Setup and parameters: We set $w_0 = \{1\}^d$ as the initial point for artificial datasets and $w_0 = \{0\}^d$ for real-world datasets. As for the step size, we use the typical $\eta = a_1/\sqrt{n}$ for SGD, $\eta = a_2$ for StGD, mini-batch SGD and mini-batch StGD (which is the default setting in our theoretical analysis) and $\eta = a_3$ for GD. The above a_1, a_2 and a_3 are all constants and we use grid search to find the best values of them for different datasets. As for the iteration times, given the training set with size n and d features, we set 500 iterations for GD and $10 * (\ln n + \ln d)$ for StGD. SGD stops iteration after a single pass over all training samples. Mini-batch StGD has batch size $n/\ln n$ and $\ln^2 n$ iterations. Mini-batch SGD has the same batch size, but $\ln n$ iterations. Finally, we set the noise parameter $\sigma = (\ln n + \ln d)/n$ for StGD and $\sigma = \ln^2 n/n$ for mini-batch StGD.

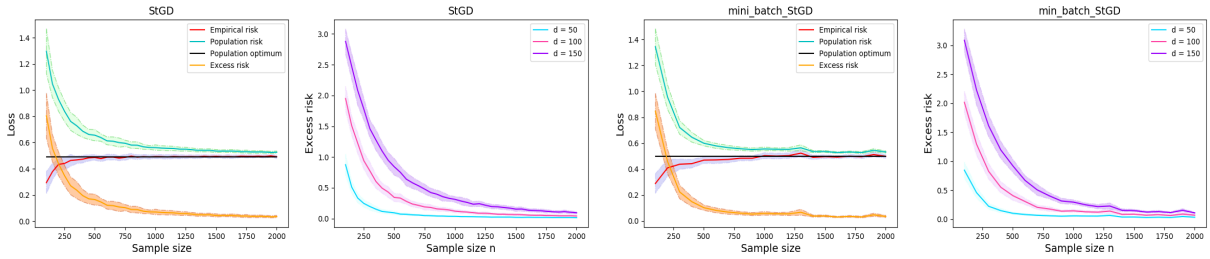
5.2 EVALUATIONS OF STGD

In the first set of experiments, we validate the theoretical promise of StGD and mini-batch SGD on artificial datasets. To show the convergence in terms of the sample size n, we sample a series of artificial datasets with size $n \in \{50, 100, 150, ..., 2000\}$ and run these algorithms on those datasets. To show how feature size d influences the convergence of StGD, we generate samples with feature size $d = 100$ and $d = 150$ and report corresponding risks. We repeat the experiment 50 times and report the mean and standard deviation of the results.

We report the population risks (test loss on the large fresh set) $F(\hat{w}_n)$, empirical risks (train loss) $\hat{F}(w_n)$, population optimum (estimated by the large fresh set) and excess risks $F(\hat{w}_n) - F(w^\star)$. Fig. 1 (a) and Fig.1 (c) illustrate the convergence rate of StGD and mini-batch SGD respectively. As n increases, the population risks of StGD and mini-batch StGD converge to population optimum. Fig. 1 (b) and (d) show how the feature size d influences the convergence rate. Larger d implies a slower convergence rate.

5.3 COMPARISON of STGD, SGD AND GD

In the second set of experiments, we compare StGD, SGD and GD on small datasets in terms of excess risk/test loss and test error rate. For the real-world datasets, we first sample 20% data points from the whole datasets to be the test set, and let the remaining samples to be the train set. Afterwards, we sample a series of datasets with size $n \in \{10, 20, ..., 250\}$ (for diabetes and breast cancer) and $n \in \{20, 40, ..., 400\}$ (for german.numer) from the remaining train set and run these algorithms on those datasets. The artificial data split is the same as the first set experiment. Given each n, we train the model and report the loss and error rate on the test set. We repeat the above procedure 10 times and report the mean and standard deviation of



(a) (b) (c) (d)

Figure 1: The StGD and mini-batch StGD on artificial data. (a) The risks of StGD. (b) The excess risks of StGD with different data dimension d. (c) The risks of mini-batch StGD. (d) The excess risks of mini-batch StGD with different data dimension d. The X-axis is the number of samples, and the Y-axis is the Risk/Loss.

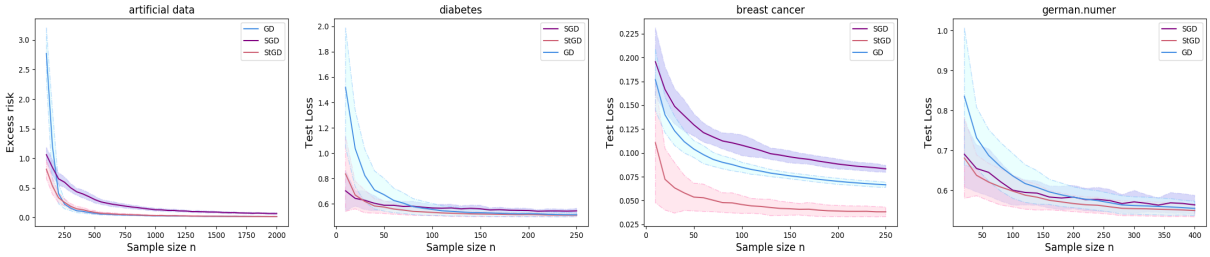


Figure 2: Compare the StGD, SGD and GD on both artificial datasets and small real-world datasets. The X-axis and the Y-axis refer to Fig. 1. The excess risk of StGD converges as fast as GD on artificial dataset. The StGD outperforms GD and SGD in terms of the test loss on real-world datasets.

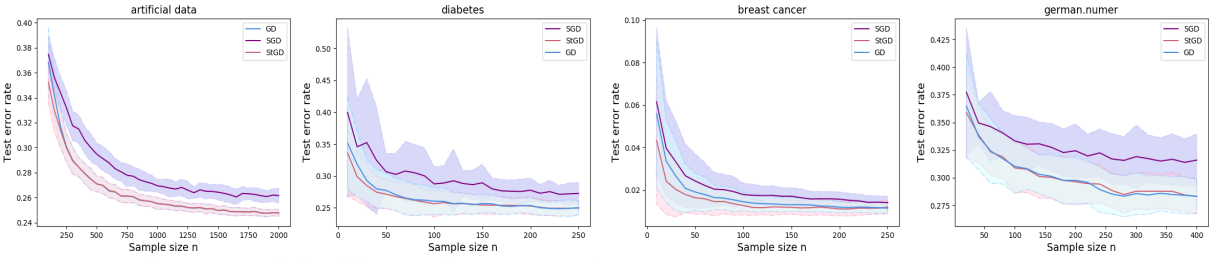


Figure 3: Compare the StGD, SGD and GD on both artificial dataset and small real-world datasets. The X-axis and the Y-axis refer to Fig. 1. The StGD outperforms GD and SGD in terms of the test error rate on artificial dataset and real-world datasets.

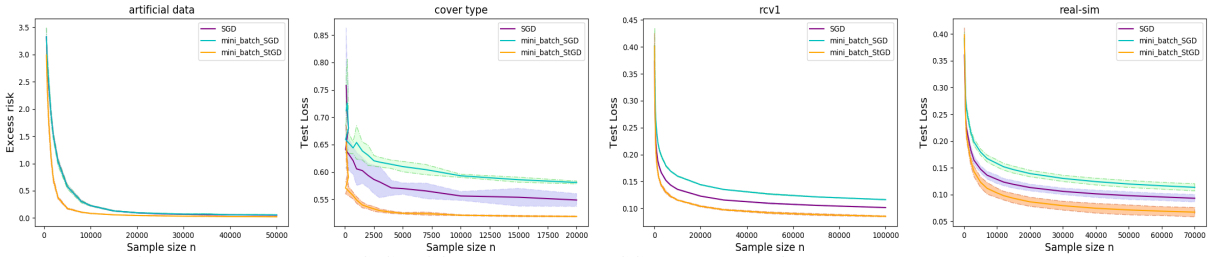


Figure 4: Compare the mini-batch StGD, SGD and mini-batch SGD on both artificial dataset and large real-world datasets. The X-axis and the Y-axis refer to Fig. 1. The excess risk of mini-batch StGD converges as fast as GD on artificial dataset. The mini-batch StGD outperforms SGD and mini-batch SGD in terms of the test loss on real-world datasets.

the results.

Fig. 2 presents the excess risks and test losses on four small datasets of the three algorithms and Fig. 3 compares the test error rates. For artificial datasets, StGD performs nearly the same as GD in terms of the excess risks and test error rates. For diabetes, breast cancer and german.numer, StGD converges better than GD. In terms of the variance, these three algorithms perform more variance on real-world data

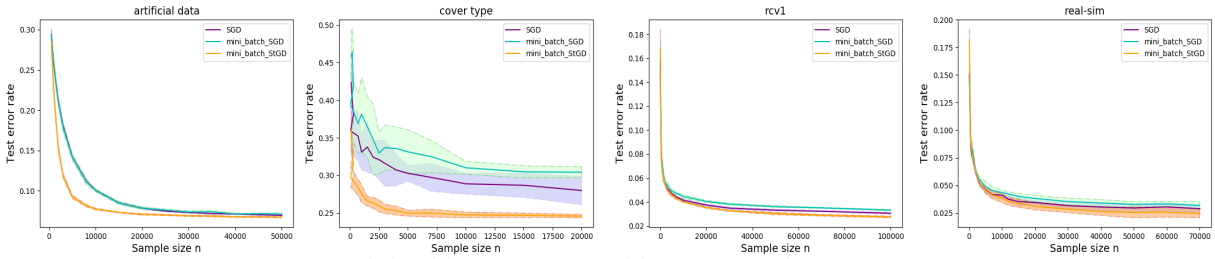


Figure 5: Compare the mini-batch StGD, SGD and mini-batch SGD on both artificial dataset and large real-world datasets. The X-axis and the Y-axis refer to Fig. 1. The test error rate of mini-batch StGD converges as fast as GD on artificial dataset. The mini-batch StGD outperforms SGD and mini-batch SGD in terms of the test error rate on real-world datasets.

than artificial data. The variance of the losses and error rates from repeated runs come form the training data and the algorithm themselves. The noise amount in StGD and the size of the training sample play an important role in variances. Repeating training the models with small dataset brings out large variance. Fig. 2 and Fig. 3 show that the variances decrease as the sample size increases.

5.4 COMPARISON of MINI-BATCH SGD, MINI-BATCH STGD AND SGD

In the third set of experiments, we compare mini-batch StGD, mini-batch SGD and SGD on large datasets. For the real-world datasets, the train and test data split is the same as the second set experiment. From the train set, we sample a series of datasets with size $n \in \{100, 300, ..., 50000\}$ for artificial data, $n \in \{100, 200, ..., 20000\}$ for cove type, $n \in \{100, 500, ..., 100000\}$ for rcv1 and $n \in \{100, 500, ..., 70000\}$ for real-sim. Given each n, we train the model and report the loss and error rate on the test set. We repeat the above procedure 10 times and report the mean and standard deviation of the results.

The excess risks on artificial dataset and the test losses on real-world datasets are shown in Fig. 4 and the test error rates is given in Fig. 5. The results show mini-batch StGD achieves the lowest test loss and test error rate for four datasets and lowest variance for cover type (three algorithms perform low variance in the other three datasets). Compared to training with small datasets (Fig.2 and Fig. 3), we observe less variance with large datasets.

6 CONCLUSION

In this paper, we study the optimization problems in machine learning. Considering the difficulty of obtaining new samples for gradient descent to approximate population gradient, we propose a stable gradient descent algorithm based on adaptive data analysis and differential privacy. We demonstrate StGD works as a basic gradient descent which has access to fresh sample at each iteration. Furthermore, we theoretically analyze that the proposed algorithm converges fast to the population optimum with high probability. Finally, we compare the proposed algorithm with existing methods in experiments. The empirical evaluation illustrate the promise of the proposed algorithm and demonstrated it outperforms existing methods.

Acknowledgements: We thank the reviewers for their valuable comments. The research was supported by NSF grants IIS-1563950, IIS-1447566, IIS-1447574, IIS-1422557, CCF-1451986, CNS-1314560, IIS-0953274, IIS-1029711, and NASA grant NNX12AQ39A.

References

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Foundations of Computer Science (FOCS)*, pages 464–473. IEEE, 2014.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

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

Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toni Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2350–2358, 2015a.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349 (6248):636–638, 2015b.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the 47th annual ACM symposium on Theory of computing (STOC)*, pages 117–126. ACM, 2015c.

Roy Frostig, Rong Ge, Sham M Kakade, and Aaron Sidford. Competing with the empirical risk minimizer in a single pass. In *Conference on learning theory (COLT)*, pages 728–763, 2015.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *Journal of Machine Learning Research*, 15(1): 2489–2512, 2014.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 449–456, 2012.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21st international conference on Machine learning (ICML)*, page 116. ACM, 2004.