Stochastic Gradient Made Stable: A Manifold Propagation Approach for Large-Scale Optimization

Yadong Mu, Member, IEEE, Wei Liu, Member, IEEE, and Wei Fan, Member, IEEE

Abstract—Stochastic gradient descent (SGD) holds as a classical method to build large scale machine learning models over big data. A stochastic gradient is typically calculated from a limited number of samples (known as mini-batch), so it potentially incurs a high variance and causes the estimated parameters bounce around the optimal solution. To improve the stability of stochastic gradient, recent years have witnessed the proposal of several semi-stochastic gradient descent algorithms, which distinguish themselves from standard SGD by incorporating global information into gradient computation. In this paper we contribute a novel stratified semi-stochastic gradient descent (S3GD) algorithm to this nascent research area, accelerating the optimization of a large family of composite convex functions. Though theoretically converging faster, prior semi-stochastic algorithms are found to suffer from high iteration complexity, which makes them even slower than SGD in practice on many datasets. In our proposed S3GD, the semi-stochastic gradient is calculated based on efficient manifold propagation, which can be numerically accomplished by sparse matrix multiplications. This way S3GD is able to generate a highly-accurate estimate of the exact gradient from each mini-batch with largely-reduced computational complexity. Theoretic analysis reveals that the proposed S3GD elegantly balances the geometric algorithmic convergence rate against the space and time complexities during the optimization. The efficacy of S3GD is also experimentally corroborated on several large-scale benchmark datasets.

Index Terms—Large-scale optimization, semi-stochastic gradient descent, manifold propagation.

1 INTRODUCTION

Regularized risk minimization [20] is a fundamental subject in machine learning and statistics, whose formulations typically admit a combination of a loss function and a regularization term. This paper addresses a general class of convex regularized risk minimization problems which can be expressed as a composition:

$$w^* = \arg\min_w \{ P(w) := P(w^\top x) + R(w) \},$$

(1)

in which $w, x$ denote the parameters and data vector respectively. Both $P(w^\top x)$ and $R(w)$ are assumed to be convex functions. Moreover, let $P(w^\top x)$ be a weighted addition of many atomic loss functions, each of which is differentiable. We simply define each atomic function on an input data pair $(x_i, y_i)$, where $x_i \in \mathbb{R}^d$ represents a feature vector and $y_i$ denotes its associated label. Popular choices of the loss functions include the square loss $(w^\top x_i - y_i)^2$, the logistic loss $\log(1 + \exp(-y_i w^\top x_i))$, and the hinge loss $\max(1 - y_i w^\top x_i, 0)$. In the above cases $y_i \in \{ \pm 1 \}$, yet in others $y_i$ can be real-valued in regression problems or missing in an unsupervised learning setting. $R(w)$ defines a proper regularization function. It imposes some structural preference on the parameters (e.g., structured sparsity or matrix low-rankness). $R(w)$ can be non-smooth with respect to $w$, such as the sparsity-encouraging 1-norm $\|w\|_1$.

When facing a large volume of training data, the space and time complexities become the critical limiting factor in building a machine learning model. In such scenarios, stochastic (sub)gradient descent (SGD) [2], [6], [9], [13], [15], [18], [24] is a favored method used by many theorists and practitioners. The most attractive trait of SGD is the lightweight computation at each iteration of updates. Its single-sample or mini-batch [5], [18] updating scheme is a general remedy for the $O(n)$ complexity in exact gradient descent (GD) methods ($n$ represents the number of training samples). Therefore, SGD algorithms are particularly promising whenever there is a limited budget of resources. Given properly-specified step size parameters at each iteration, SGD algorithms often enjoy provably rates of convergence.

The major downside of SGD in practical implementations is caused by large gradient variance. Statistically, the mathematical expectation of stochastic gradient is exactly the full gradient. However, the randomness in constructing mini-batch brings large variance to stochastic gradients, particularly for complex data set. In other words, moving along the direction of stochastic gradient does not always guarantee a decrease of the entire training loss. Under large gradient variance, the estimated parameters often drastically bounce around the global optimal solution.

Recent years have witnessed the emerging efforts of developing sophisticated algorithms which reduce the gradient variance in SGD. The shared idea underlying these works is incorporating an additional gradient-correcting
operation when computing the stochastic gradient. The corrected stochastic gradient becomes a more accurate approximation of the full gradient. Statistically, it enjoys a reduced level of variance. For example, the work in [28] explicitly expresses the gradient variance and proves that constructing mini-batch using special non-uniform sampling strategy is able to reduce the gradient variance. The sampling probability is essentially based on the contextual importance of a sample. Another method named stochastic average gradient (SAG) [17] keeps a record of historic stochastic gradients and adaptively averages them for the use in the current iteration. The rate of convergence is thereby improved to \( O(1/k) \) for general convex functions, and \( O(p^k) \) with \( p < 1 \) for strongly convex functions, respectively \((k\) is the count of iterations). However, storing historic gradients in SAG entails a heavy burden for machine learning models with many parameters.

This paper advocates an efficient manifold propagation approach for reducing stochastic gradient variance in large-scale machine learning. It aims to improve the stability of the stochastic gradient, such that large descending step sizes can be used for faster convergence. We adopt the computational framework of residual-minimizing gradient correction which was originally proposed in stochastic variance-reduced gradient (SVRG) [10] by Johnson and Zhang. The computational framework is comprised of two steps: 1) estimate the residual between a stochastic gradient and the full gradient using global information, and 2) compensate the stochastic gradient such that the residual is minimized towards 0.

Since the optimization proceeds in rounds, we can thus describe it with an update rule. Assume \( w^k \) is the latest estimation for the problem \( \min_w F(w) \) at the \( k \)-th iteration, standard SGD and full (sub)gradient descend (GD) will seek for a new estimation \( w^{k+1} \) according to 1:

\[
\text{(SGD):} \quad w^{k+1} = w^k - \eta_k \nabla F_i(w^k),
\]
\[
\text{(GD):} \quad w^{k+1} = w^k - \eta_k \nabla F(w^k),
\]

1. When \( F(w) \) is non-smooth, sub-differential (rather than gradient) will be used. However, we here abuse the notation \( \nabla \) for statement conciseness.

where \( \eta_k \) is a delicately-chosen step-size parameter. For notational simplicity, we here assume that each mini-batch only contains a single random sample. The term \( F_i(w^k) \) in SGD denotes the atomic function conditioned on a random sample \( x_i \) and the latest parameters \( w^k \). \( F(w^k) \) is computed using all training set.

In contrast, semi-stochastic gradient is obtained by the rule below:

\[
w^{k+1} = w^k - \eta_k \left( \nabla F_i(w^k) - \left( \nabla F_i(\bar{w}) - \nabla F(\bar{w}) \right) \right),
\]

where \( \bar{w} \) represents some historic memory of recent parameter estimation. \( \bar{w} \) is supposed to be proximal to \( w^k \). The term \( \nabla F_i(\bar{w}) - \nabla F(\bar{w}) \) approximately estimates the residual between the stochastic gradient of sample \( x_i \) and full gradient. By subtracting the residual term from \( \nabla F_i(w^k) \), it naturally aligns the stochastic gradient with the full gradient. As an extreme case, letting \( \bar{w} = w^k \) will immediately get the full gradient in (4). The idea is intuitively explained in Figure 1.

Theoretic analysis in [10], [11], [22] reveals that semi-stochastic algorithms achieve a geometric rate of convergence. Though such a convergence rate is generally regarded as the synonym of satisfactory efficiency, it is important to emphasize that this rate is achieved at the cost of higher iteration complexity compared to standard SGD. In our experiments, we are surprised to find that SGD still dominates in many cases, since its light-weight iteration cost compensates its slow theoretic convergence rate. In other words, the promising geometric convergence rate of existing semi-stochastic algorithms is probably Pyrrhic victories at excessive costs of maintaining high-accuracy estimation of gradient residual.

We find that a comprehensive quantitative comparison between semi-stochastic algorithms and SGD is still missing in the literature. In fact, most existing semi-stochastic algorithms either rely on periodic full gradient computation [10] or use Hessian-like covariance matrix operations [21], which account for their high iteration complexities. In this paper we expose a novel way of efficiently computing semi-stochastic gradient and evaluate it on a variety of massive data sets. We term the new method as stratified semi-stochastic gradient descent (S3GD) hereafter. Our major contributions are described below:

- As a crucial component of the proposed S3GD, we devise an efficient manifold propagation approach for computing semi-stochastic gradient. First, a fixed number of anchors are drawn in a stratified manner. After that, each sample in the training set is connected to its adjacent anchors, forming a graph-defined manifold. At each iteration, the gradient information computed on the anchors diffuses over the manifold, obtaining an approximate estimation of the full gradient. The idea empirically proves to be a strong competitor to the existing expensive, albeit accurate, gradient-correcting operations such as SVRG.

- We provide theoretic analysis about S3GD. Under standard assumptions imposed on the objective functions (i.e., strong convexity and Lipschitz continuity) and with a constant step size, S3GD achieves a
geometric convergence rate (in terms of parameter optimality and iterates) up to a constant which is essentially determined by the Laplacian matrix of manifold-induced graph.

- Last but not least, we conduct quantitative investigation over 9 different benchmarks, covering a large spectrum of real-world problems. The experimental evaluations fully validate the efficiency and effectiveness that S3GD brings. Moreover, the comparisons between various semi-stochastic algorithms and classic SGD is so far the most comprehensive and supposed to be very useful for re-calibrate the research direction of semi-stochastic algorithms.

The remainder of this paper is organized as follows: We start in Section 2 by describing preliminary knowledge and algorithmic details of S3GD. Specifically, Section 2.4 is devoted to applying the generic idea of S3GD to several representative machine learning problems. Then we give the theoretic analysis in Section 3, where the major observation is found in Theorem 3.1. In Section 4 we present the quantitative investigation of S3GD on several large-scale benchmark datasets widely used in machine learning and statistics. Finally, in Section 5 we draw the concluding remarks and discuss the future perspective.

2 THE PROPOSED ALGORITHM

2.1 Notations and Assumptions

Notations: We will denote vectors and matrices by boldface letters. Let $||x||_2, ||x||_1$ be the Euclidean norm and 1-norm (summation of all absolute elements) of a vector respectively. Denote the training data set as $X = \{(x_i, y_i)\}$, which has a cardinality of $n$ and $i$ therein represents the index. Each sample is described by a tuple $(x_i, y_i)$, where the feature vector $x_i \in \mathbb{R}^d$ and $y_i$ corresponds to either labels in supervised learning or response values in regression problems.

The smooth part in Problem (1) is posed in an additive form, namely $P(w) = (1/n) \sum_{i=1}^{n} \psi(x_i,y_i,w)^2$. The regularization term $R(w)$ is convex yet not mandatorily differentiable. Whenever not incurring confusion, we use the notation $\psi_i(w)$ for simplifying $\psi(x_i,y_i,w)$. $|x|_+ = \max(x,0)$ is the zero-thresholding operation.

Our theoretic observations are based on the following assumptions, similar to previous semi-stochastic gradient descent methods [22], [27]:

**Assumption 2.1. (strong convexity):** We say that a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is strongly convex, if there exists $\mu > 0$ such that for all $u, v \in \mathbb{R}^d$,

$$f(u) \geq f(v) + \xi^T (u - v) + \frac{\mu}{2} ||u - v||^2, \forall \xi \in \partial f(v), \tag{5}$$

where $\partial f(v)$ is the sub-differential (set of sub-gradients) at point $v$. The convexity parameter is defined to be the largest $\mu$ that satisfies the above condition. Let $P(w), R(w)$ and their composition $F(w)$ have non-negative convexity parameters $\mu_P, \mu_R$ and $\mu$ respectively. It is easily verified that $\mu \geq \mu_P + \mu_R$ by definition of strong convexity and function composition.

**Assumption 2.2. (smoothness):** A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is $L$-smooth if it is differential and there exists $L > 0$ such that it satisfies

$$f(u) \leq f(v) + \nabla f(v)(u - v) + \frac{L}{2} ||u - v||^2, \tag{6}$$

for all $u, v \in \mathbb{R}^d$. Or equivalently, its gradient is $L$-Lipschitz continuous, namely we have

$$||\nabla f(u) - \nabla f(v)|| \leq L ||u - v||. \tag{7}$$

Let the Lipschitz parameter for each atomic function $\psi_i(w)$ be $L_i$ respectively. The Lipschitz parameter for their composition $P(w)$ is $L_P \leq (1/n) \sum_{i=1}^{n} L_i$. The regularization term $R(w)$ is mostly assumed to be non-differentiable and thus has no Lipschitz parameter.

2.2 Algorithmic Framework

The composite optimization problem in (1) is of broad interests in machine learning and data mining fields. Nonetheless, solving it at optimal convergence speed is non-trivial. If we simply treat $F(w)$ as a black-box oracle which only returns the first-order (sub)gradient, there are several off-theshelf tools, including SGD and full (sub)gradient descent. Since full (sub)gradient estimation is extremely expensive when huge volume of data is available, recent work has focused on stochastic optimization.

SVRG [10], as introduced in preceding section, obeys the update rule in (4). Procedurally, it utilizes two nested loops. At each iteration of the outer loop, it memorizes a recent...
estimation $\tilde{w}$ and calculates the full gradient $\nabla F(\tilde{w})$ at $\tilde{w}$. In the inner loops, it calculates $\nabla F_i(w^k)$ and $\nabla F_i(\tilde{w})$ for mini-batches, and afterwards amends the stochastic gradient $\nabla F_i(w^k)$ by the rule in (4). Note that the same $\tilde{w}$ is used for all update within an outer loop. The SVRG method, though simple, profoundly reduces the amortized time complexity at iterations and theoretically achieves geometric convergence rate for strongly-convex smooth functions.

Another semi-stochastic algorithm, stochastic control variate (SCV) [21], represents a general approach of using control variate for variance reduction in stochastic gradient. The update rule of SCV is similar to (4) yet the last two (sub)gradients in (4) are replaced by control variate. Data statistics such as low-order moments (vector mean and covariance matrix) is used to form the control variate. The authors apply SCV to solve logistic regression and latent Dirichlet allocation.

However, existing semi-stochastic methods like SVRG and SCV are not guaranteed to beat standard SGD in practice, since computing $\nabla F(\tilde{w})$ in SVRG or control variate in SCV significantly increases the iteration complexity. To overcome the key limitations that dramatically restrict their capability in large scale data analysis, we propose S3GD. Algorithm 1 sketches the pseudo-code of S3GD.

Before diving into algorithmic details, we want to highlight two defining traits of S3GD:

**Manifold-oriented gradient approximation:** Given the composite function $F(w)$, S3GD only computes the gradient on the smooth part $P(w)$. For accelerating the computation of semi-stochastic gradient in (4), we argue that the key is to find a function $H(w)$, whose design principals are:

1. $H(w)$ is a good surrogate to $\nabla P(w)$, namely $H(w) \approx \nabla P(w)$;
2. $H(w)$ can be efficiently computed;
3. $H(w)$ is additive, namely $H(w) = \sum_{i=1}^{n} h_i(w)$, where $h_i(w) \approx \nabla \psi_i(w)$ approximates the gradient of an atomic function defined on $x_i$.

We defer the construction of function $H(w)$ in Section 2.3, focusing on the algorithmic pipeline here. At specific iteration, denote the index set of a random mini-batch as $\mathcal{I}$. Conditioned on current parameter estimation $w^k$, the semi-stochastic gradient in S3GD is computed by the following formula:

$$g_{\mathcal{I}}(w^k) = \nabla \psi_{\mathcal{I}}(w^k) - [h_{\mathcal{I}}(w^k) - H(w^k)],$$

where $h_{\mathcal{I}}(w^k) = \sum_{i \in \mathcal{I}} h_i(w^k)/|\mathcal{I}|$ is the averaged approximate stochastic gradient over index set $\mathcal{I}$.

$g_{\mathcal{I}}(w^k)$ actually provides an unbiased estimate of $\nabla P(w^k)$ when $\mathcal{I}$ is uniformly drawn from the index set $[1, \ldots, n]$ with replacement. Its soundness is naturally fulfilled by the additive construction of function $H$. Consequently, the variance of $g_{\mathcal{I}}(w^k)$ becomes

$$Var\left[g_{\mathcal{I}}(w^k)\right] = \mathbb{E}\left\|\nabla \psi_{\mathcal{I}}(w^k) - h_{\mathcal{I}}(w^k)\right\|^2 - \mathbb{E}\left\|\nabla P(w^k) - H(w^k)\right\|^2 \
\leq \mathbb{E}\left\|\nabla \psi_{\mathcal{I}}(w^k) - h_{\mathcal{I}}(w^k)\right\|^2$$

Intuitively, the relative comparison of $Var[g_{\mathcal{I}}(w^k)]$ and $Var[g_{\mathcal{I}}(w^k)]$ essentially hinges on which of $h_{\mathcal{I}}(w^k)$ or $\nabla P(w^k)$ is more close to $\nabla \psi_{\mathcal{I}}(w^k)$. As shown later, we tailor $h_{\mathcal{I}}(w^k)$ to be a local approximation to $\nabla \psi_{\mathcal{I}}(w^k)$, which is supposedly superior to the global average $\nabla P(w^k)$, particularly when the input data set is with rich variety.

**Proximity-regularized linear approximation:** After the semi-stochastic gradient $g_{\mathcal{I}}(w^k)$ is computed, we further solve the following sub-problem:

$$w^{k+1} = \arg \min_w P(w^k) + \langle g_{\mathcal{I}}(w^k), w - w^k \rangle - \frac{1}{2\eta} \|w - w^k\|^2 + R(w),$$

where the first three terms define a proximal regularization of the linearized approximation of $P(w)$ around point $w^k$. $R(w)$ is presumably in a good shape such that solving (12) is trivial. If $R(w)$ is itself composition of several non-smooth functions, one can resort to the modern proximal average techniques [1], [30]. Moreover, it is verified that Problem (12) can be compactly abstracted by the operation $\text{prox}_R(u)$ below:

$$\text{prox}_{\eta R}(u) = \arg \min_u \frac{1}{2} \|w - u\|^2 + \eta R(w),$$

where $u = w^k - \eta \cdot g_{\mathcal{I}}(w^k)$.

### 2.3 Gradient Approximation by Manifold Propagation

This section elaborates on a manifold-oriented method for approximating the gradients $\nabla \psi_i(w)$ and $\nabla P(w)$. Our key argument is that a universal gradient-approximating function is either infeasible or inaccurate in general, particularly when the loss function $P(w)$ has large condition number $L_P/\mu_P$. Our proposed remedy is anchor-based gradient approximation over data manifold. The idea has ever been explored in other context yet not in stochastic optimization before. For example, in [23] Yu et al. showed that any Lipschitz-continuous function $f(x)$ residing on lower-dimensional manifolds can be approximated by a linear combination of function values, namely

$$f(x) = \sum_{z \in Z} \gamma_z(x) f(z),$$

where $Z$ is a collection of pre-specified anchors. $\gamma_z(x) \geq 0$ is the combination coefficient depending on both $x$ and anchor $z$. The idea is later generalized in the work of locally-linear support vector machine [8], [12], where each anchor determines a function (rather than a fixed value), namely $f(z)$ in (16) is replaced by an $x$-varying function $f_x(x)$.

Recall that in Problem (1), the loss function $P()$ is defined on $w \cdot x$. Letting $\psi'(u)$ be the derivative with respect to scalar $u$, the gradient of $P(w)$ can be factorized as below:

$$\nabla P(w) = (1/n) \sum_{i=1\ldots n} \nabla_w \psi(w^T x_i) = (1/n) \sum_{i=1\ldots n} \psi'(w^T x_i) \cdot x_i.$$
Algorithm 2 Manifold Based Gradient Approximation

1: Parameters: anchor number $m$ and k-NN parameter $k$.

2: Anchor Selection
3: Perform data clustering to obtain $m$ centers $c_i, i = 1 \ldots m$;
4: for $i = 1$ to $m$ do
5: Find anchor $z_i$ by solving

\[ z_i = \arg \min_{x} \|x - c_i\|^2, \quad (14) \]

where $x$ is from the training data set.
6: end for

7: Sparse Anchor-Sample Graph (ASG) Construction
8: for $i = 1$ to $n$ do
9: For sample $x_i$, find $k$-nearest anchors $z_{i1}, \ldots, z_{ik}$;
10: Learn the Gaussian kernel parameter by

\[ \sigma = \max \left( \epsilon, \inf_{j \in (i_1, \ldots, i_k)} \sqrt{\|x_i - z_j\|} \right), \quad (15) \]

where $\epsilon$ is set to be $10^{-4}$ to avoid the trivial case $\sigma = 0$.
11: for each $k$-nearest anchor $z$ do
12: Calculate $\gamma_a(x_i) = \exp(-\|x_i - z\|^2/2\sigma^2)$;
13: end for
14: Normalize $\gamma_a(x_i)$ to ensure that they sum to 1;
15: end for

16: Gradient Approximation over ASG
17: Pre-compute the product matrix $XM$ in Eqn. 23;
18: for approximate gradient $h_{\psi}(w)$ of any mini-batch
19: for each $z_i$ in the mini-batch $I$ do
20: Calculate $h_i(w)$ by Eqn. 20;
21: end for
22: for all $z \in Z$
23: $h_{\psi}(w) = \sum_{z \in Z} h_i(w)/|Z|$;
24: for approximate full gradient $H(w)$
25: Calculate $H(w)$ by Eqn. 23;

Inspired by the factorization in (17), we propose to establish a manifold over the training data, such that the derivative term $\psi'(w^T x_i)$ in (17) can be efficiently computed via sparse information propagation on the manifold. Algorithm 2 shows the pseudo-code for the major steps. The proposed scheme consists of the following components:

1) Constructing anchor set: Compared to universal gradient approximation, anchor set [14] has a stronger representation power by establishing locally low-order (such as quadratic or cubic) approximation around each anchor point. Let $m$ be the number of anchor points. The optimal value of $m$ is mostly dataset-specific. Let $Z = \{z_1, \ldots, z_m\}$ be the anchor set. We employ a k-means clustering procedure to obtain $m$ centers in a stratified manner. The anchor points are chosen as the nearest samples to these centers, since these centers per se are not necessarily corresponding to meaningful features.

2) Anchor-Sample Graph (ASG) Construction: We follow the local approximation scheme as described in Eqn. (16). Let each anchor $z$ uniquely determine a localized function $f(z)$. Since the primary goal is to approximate the gradient, let us define $f(z) = \psi'(w^T z)$. Moreover, let us assume that the derivative $\psi'(w^T x)$ for any $x$ lies on a non-linear manifold, namely

\[ \psi'(w^T x) \approx \sum_{z \in Z} \gamma_a(x_i) \cdot \psi'(w^T z). \quad (18) \]

Obviously, anchors and samples naturally form an anchor-sample graph (ASG), where the connectivity strengths are controlled by $\{\gamma_a(x_i)\}$. In graph-based propagation methods, it is known that connecting sample with remote anchors potential does harm to the performance [14]. Therefore, each sample is enforced to only connect to its $k$-nearest anchors. The computation of $\gamma_a(x)$ is detailed in Algorithm 2.

3) Gradient Approximation over ASG: Based on the factorization in Eqn. (17), the stochastic gradient for a sample $x_i$ can be computed by:

\[ \nabla \psi'(w^T x_i) = \psi'(w^T x_i) \cdot x_i \approx (\sum_{z \in Z} \gamma_a(x_i) \cdot \psi'(w^T z)) \cdot x_i, \quad (19) \]

where the last approximation holds from the manifold assumption as described in Eqn. (18). The right hand side in (19) serves as our proposed manifold-oriented approximate gradient, namely

\[ h_i(w) = \left( \sum_{z \in Z} \gamma_a(x_i) \cdot \psi'(w^T z) \right) x_i. \quad (20) \]

Importantly, computing (20) is highly efficient owing to the sparsity of ASG. It only involves executing the derivative function for all anchors in addition to another $O(k + d)$ algorithmic operations per sample. Likewise, the approximate full gradient $H(w)$ in (9) can be computed by:

\[ H(w) = \frac{1}{n} \sum_{i=1}^{n} h_i(w) = \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{z \in Z} \gamma_a(x_i) \cdot \psi'(w^T z) \right) x_i \]

\[ = \frac{1}{n} \sum_{z \in Z} \sum_{i=1}^{n} \gamma_a(x_i) \cdot x_i \cdot \psi'(w^T z), \quad (21) \]

which is purely based on the sparse edge coefficients in ASG and the derivatives $\{\psi'(w^T z)\}$ for all anchor $z$.

In fact, the computation in (21) can be further accelerated by per-computing the terms irrelevant to $w$. Let $M \in \mathbb{R}^{n \times m}$ be the matrix by compiling all coefficients in ASG. Specifically, $M(i, j) = \gamma_a(x_i)$. Moreover, let

\[ g_z(w) = (\psi'(w^T z_1), \ldots, \psi'(w^T z_m))^T \in \mathbb{R}^m, \quad (22) \]

be the vector of anchor derivatives conditioned on parameter $w$ and $Z = \{x_1, \ldots, x_n\} \in \mathbb{R}^{d \times m}$ be the feature matrix. Eqn. (21) can be compactly expressed as

\[ H(w) = \frac{1}{n} \cdot (XM) \times g_z(w), \quad (23) \]

The product $XM \in \mathbb{R}^{d \times m}$ is not varying with respect to $w$ and thus can be pre-computed for avoiding redundant computation at different outer loops in Algorithm 1.

2.4 Instances of Applications

This section instantiates our proposed algorithm by several representative loss functions and regularizations.

Logistic Loss: It is applicable to either real or binary responses. We focus on the binary case, where $y = \pm 1$. For any data vector $x$, the conditional probability of the class label is:

\[ p(y|x; w) = \sigma(yw^T x) := 1/(1 + \exp(-y(w^T x))) \quad (24) \]

3. For the sake of simplifying notations, we remove the intercept variable by appending an additional dimension of constant 1 to any feature vector $x$. 

\[ \]
The log-likelihood function is then expressed as \( P(w) = \sum_{i=1}^{n} \psi(w^\top x_i) = \sum_{i=1}^{n} \log p(y_i|x_i; w) \). According to the calculus rule of sigmoid function, the gradient of \( \psi(w^\top x_i) \) is:

\[
\nabla \psi(w^\top x_i) = \sigma(-y_i w^\top x_i) \cdot y_i \cdot x_i
\]

(25)

Directly plugging the derivative into Eqn. (20) is problematic, since the label \( y_i \) and feature vector \( x_i \) are tightly coupled in Eqn. (25). Therefore, the stochastic gradient of a sample \( x_i \) shall be handled according to its label. More formally, let us consider the following two cases:

Case-1: \( y_i = 1 \). We have \( \nabla \psi(w^\top x_i) = \sigma(-w^\top x_i) \cdot x_i \), which can be efficiently solved by the tricks developed in Eqn. (20).

Case-2: \( y_i = -1 \). Now there is \( \nabla \psi(w^\top x_i) = \sigma(w^\top x_i) \cdot (x_i) = (1 - \sigma(-w^\top x_i)) \cdot x_i = x_i + \sigma(-w^\top x_i) \cdot x_i \).

Note that we use the property of sigmoid function \( \sigma(u) = 1 - \sigma(-u) \). It turns out that we can still apply the tricks in Eqn. (20) by amending the result with the term \(-x_i\).

The matrix representation in Eqn. (23) shall be accordingly adjusted. However we omit it and leave it to longer version of this paper.

Hinge Loss and Squared Hinge Loss: The loss function popularized by SVM is known to be hinge loss \( |yw^\top x|_+ \). It is non-differentiable due to the irregularity at \( yw^\top x = 1 \). However, as discovered in [25], hinge loss can be smoothed by the loss of "modified logistic regression":

\[
[1 - yw^\top x]_+ \approx \frac{1}{\beta} \log \left( 1 + \exp(-\beta(yw^\top x - 1)) \right).
\]

(26)

The approximation residual asymptotically becomes zero when \( \beta \to +\infty \), therefore we can cast hinge loss into the framework of logistic loss with properly-chosen \( \beta \).

Another solution of smoothing hinge loss is using squared hinge loss as adopted by L2-SVM [4], namely \((1/2)[|y(w^\top x)|_+]^2\), which naturally removes the irregular point at the risk of over-penalizing large response. Its gradient at a sample \((x_i, y_i)\) is

\[
\nabla \psi(w^\top x_i) = [1 - y_i w^\top x_i]_+ \cdot (y_i) \cdot x_i.
\]

(27)

To decouple \( x_i \) and \( y_i \), we can apply the technique which was just discussed about Eqn. (25).

Regularization: The regularization function \( R(w) \) can be either smooth (e.g., Tikhonov regularization) or non-smooth (e.g., 1-norm regularization). Below we list a few regularization functions widely-used in machine learning:

- \((\text{Tikhonov})\) : \( R(w) = \lambda \|w\|_2^2 \).
- \((1\text{-norm})\) : \( R(w) = \lambda \|w\|_1 \).
- \((\text{Elastic net})\) : \( R(w) = \lambda(1 - \alpha) \|w\|_1 + \lambda \alpha \|w\|_2 \).

When parameters \( w \) constitute a matrix rather than a vector, regularization terms such as matrix nuclear norm [3] can be applied. However, optimizing with all above regularization under the proximal operator in (13) has been maturely developed. We thus omit more discussion.

2.5 Algorithmic Complexity

The iteration complexity of the proposed S3GD depends on several variables: the mini-batch size \( p \), the number of anchors \( m \), the k-NN parameter in constructing ASG, the maximal inner loop count \( k_{in} \) and the feature dimensionality \( d \). The major computational overhead stems from computing \( H(\hat{w}) \) at each outer loop in Algorithm 1. Importantly, the compact matrix form in Eqn. (23) largely reduces the time and space complexities.

Most of existing semi-stochastic algorithms rely on two nested loop, of which the outer loop incurs exact full gradient computation or covariance matrix estimation. For large data, it entails a tremendous \( O(nd) \) or \( O(d^2) \) complexity. For other sophisticated algorithms that target at improved mini-batch construction (such as SSAMG [27]), the iteration complexity is generally better than ours. However, the lack of global information makes these algorithms more vulnerable to gradient noises.

Regarding space requirement, the major cost comes from the storage of the product matrix in Eqn. (23). To maintain such data structure, it consumes \( O(nk) \), where \( k \) is the anchor k-NN parameter. Akin to SVRG and SCV, S3GD does not memorize historic gradients. We summarize the space and time complexities for all interested algorithms in Table 1.

3 Convergence Analysis

We need two lemmas as below to forward the convergence analysis. The first lemma states that the proximal mapping is firmly nonexpansive, or co-coercive with constant not greater than 1:

**Lemma 3.1.** Let \( R(w) : \mathbb{R}^d \mapsto \mathbb{R} \) be a closed convex function with strong convexity parameter \( \mu_R \geq 0 \), then for any \( u, v \in \text{dom}(R) \) and \( \eta > 0 \),

\[
\|\text{prox}_{\eta R}(u) - \text{prox}_{\eta R}(v)\| \leq \frac{1}{1 + \eta \mu_R} \|u - v\|.
\]

(28)

The other lemma is a generalization of Theorem 2.1.5 in [16]. The proofs of both lemmas are deferred to the Appendix.

**Lemma 3.2.** Suppose \( P(w) : \mathbb{R}^d \mapsto \mathbb{R} \) is \( \mu_P \)-strongly convex and \( L_P \)-smooth, then for any \( u, v \in \text{dom}(P) \),

\[
\|\nabla P(u) - \nabla P(v)\| \leq \frac{\mu_P L_P}{\mu_P + L_P} \|P(u) - P(v)\| + \frac{1}{\mu_P + L_P} \|u - v\|^2.
\]

(29)

The following is our main observation regarding the convergence property of the proposed S3GD:

**Theorem 3.1.** For compositional function \( L(w) = P(w) + R(w) \), assume its two components \( P(w), \) \( R(w) \) have strong convexity parameter \( \mu_P \geq 0, \mu_R \geq 0 \) and \( \mu_P \cdot \mu_R > 0 \). \( P(w) \) has Lipschitz parameter \( L_P \). Set the step size \( \eta_k = \eta \in (0, \frac{2}{\mu_P + \mu_R}) \) for all \( k > 0 \). The proposed algorithm will satisfy the following inequality for the \( k \)-th iteration,

\[
\|w^k - w^*\|^2 \leq \rho^k \left( \|w^0 - w^*\|^2 - \frac{\Delta}{\rho} \right) + \frac{1}{\rho^k} \eta \Delta,
\]

where \( \rho = \frac{1}{1 + \eta \mu_R} \left( 1 - \frac{2 \mu_P \mu_R}{\mu_P + \mu_R} \right), \gamma = \frac{\eta \mu_R}{1 + \eta \mu_R} \) and \( \Delta \) reflects the maximal gradient approximation residual.
where the two equalities hold since continuity. There is also
\[ E \] where the first equality comes from the fact that \( w \) is a point, namely
\[ E \] is the number of anchors, the number of training samples, and the feature dimensionality respectively. \( k \) denotes the anchor \( k \)-NN parameter. Note that for SVRG and S3GD, they both adopt nested loop during the optimization. \( k_{in} \) denotes the maximal iteration count of the inner loop.

**Proof.** For convenience let us define the following notations at the \( k \)-th iteration
\[
\begin{align*}
v_k &= \nabla \psi_{h_k}(w^{k-1}) - [h_k(\widehat{w}) - E h_k(\widehat{w})], \\
w_k &= \text{prox}_{\eta_k R}(w^{k-1} - \eta_k v_k), \\
\Delta_k &= E(\|\nabla \psi_{h_k}(w^{k-1}) - h_k(\widehat{w})\|^2).
\end{align*}
\]
We proceed the analysis by measuring the expectation of the distance between \( w^k \) and the global optimum \( w^* \). Taking expectation with respect to the random sample index \( i_k \) there is
\[
\begin{align*}
E[\|w^k - w^*\|^2] &= E[\|\text{prox}_{\eta_k R}(w^{k-1} - \eta_k v_k) - \text{prox}_{\eta_k R}(w^* - \eta_k \nabla P(w^*))\|^2] \\
&\leq \frac{1}{1 + \eta_k \rho} \cdot \left( E(\|w^{k-1} - \eta_k \nabla P(w^*)\|^2) - E(\|w^* - \eta_k \nabla P(w^*)\|^2) \right) \\
&= \frac{1}{1 + \eta_k \rho} \|w^{k-1} - w^*\|^2 + \frac{\eta_k^2}{1 + \eta_k \rho} \cdot \left( E(\|v_k - \nabla P(w^*)\|^2) \right).
\end{align*}
\]
where the first equality comes from the fact that \( w^* \) is a fixed point, namely \( w^* = \text{prox}_{\eta_k R}(w^* - \eta_k \nabla P(w^*)) \). The first inequality is obtained by applying Lemma 3.1. Recall that \( E v_k = \nabla P(w^{k-1}) \), we have
\[
\begin{align*}
E(\|v_k - \nabla P(w^*)\|^2) &= E(\|v_k - \nabla P(w^{k-1})\|^2 + \|\nabla P(w^{k-1}) - \nabla P(w^*)\|^2) \\
&= E(\|\nabla \psi_{h_k}(w^{k-1}) - h_k(\widehat{w})\|^2) \\
&\quad - E(\|\nabla P(w^{k-1}) - E h_k(\widehat{w})\|^2) \\
&\quad + E(\|\nabla P(w^{k-1}) - \nabla P(w^*)\|^2) \\
&\leq \Delta_k + E(\|\nabla P(w^{k-1}) - \nabla P(w^*)\|^2),
\end{align*}
\]
where the two equalities hold since \( E (\|x\|^2) = E (\|Ex\|^2 + E (x - Ex)^2) \). The last inequality is from the definition of Lipschitz continuity. There is also
\[
\begin{align*}
E(v_k - \nabla P(w^*))^T(w^{k-1} - w^*) &= (\nabla P(w^{k-1}) - \nabla P(w^*))^T(w^{k-1} - w^*) \\
&\geq \frac{\mu_p + L_p}{\mu_p + L_p} \|\nabla P(w^{k-1}) - \nabla P(w^*)\|^2 \\
&\quad + \frac{\mu_p L_p}{\mu_p + L_p} \|w^{k-1} - w^*\|^2.
\end{align*}
\]
Plugging (30)(31) into (29) obtains
\[
\begin{align*}
E[\|w^k - w^*\|^2] &= \frac{1}{1 + \eta_k \rho} \left( 1 - \frac{2 \eta_k \mu_p L_p}{\mu_p + L_p} \right) \|w^{k-1} - w^*\|^2 \\
&\quad + \frac{2 \eta_k^2}{1 + \eta_k \rho} \|\nabla P(w^{k-1}) - \nabla P(w^*)\|^2 \\
&\quad + \frac{\eta_k^2}{1 + \eta_k \rho} \Delta_k
\end{align*}
\]
Set \( \eta_k = \eta \in (0, 2/(\mu_p + L_p)) \) and assume there exists \( \Delta \geq \Delta_k \) for all \( k \). We have
\[
E[\|w^k - w^*\|^2] \leq \rho^k \|w^0 - w^*\|^2 + \gamma \Delta
\]
Taking expectation with respect to all historical choice of \( i_1, \ldots, i_k \) and iteratively applying (33), we have
\[
E[\|w^k - w^*\|^2] \leq \rho^k \|w^0 - w^*\|^2 + \sum_{t=0}^{k-1} \rho^t \gamma \Delta
\]
It completes the proof. \( \square \)

**Remarks:** The above theorem basically states that, when \( P(w), R(w) \) are not simultaneously \( \delta \)-strongly convex, the upper bound in term of solution optimality is comprised of two terms. One of them exponentially converges to zero, and the other is pertaining to the accuracy of stratified manifold-based gradient approximation. Though the accuracy of gradient approximation is not amenable for bound analysis, we empirically investigates its effect in large-scale optimization. For non-strongly convex functions, adding quadratic perturbation terms can be used to reach similar argument [29]. We omit the details due to trivialness.

## 4 Experiments

This section reports the numerical studies between our proposed S3GD and other competing algorithms.

### 4.1 Description of Dataset and Applications

To make the experiments comprehensive, we include nine benchmarks that cover a variety of heterogeneous tasks and different data scales: 20-NewsGroups\textsuperscript{4} which contains nicely-organized documents from 20 different news topics, WebSpan\textsuperscript{5} represents a large collection of annotated spam or non-spam hosts labeled by a group of volunteers, IJCNN\textsuperscript{6} for time-series data, KDD04\_bio and KDD04\_phy\textsuperscript{7} which correspond to the protein homology sub-task and quantum physics sub-task in KDD-Cup 2004 respectively, covtype\textsuperscript{8} which includes cartographic variables for predicting forest cover type. We also include three computer vision benchmarks: CIFAR10\textsuperscript{9} for image categorization, Kaggle-Face\textsuperscript{10} for

4. http://qwone.com/∼jason/20Newsgroups/
5. http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/binary.html
6. http://www.geocities.com/jcnn/nnc_ijcnn01.pdf
7. http://osmot.cs.cornell.edu/kddcup/datasets.html
8. https://archive.ics.uci.edu/ml/datasets/Covertype
9. http://www.cs.toronto.edu/~kriz/cifar.html
10. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge
facial expression recognition and MED11 for video event detection.

Table 2 summarizes the critical information for above-mentioned benchmarks. For most datasets, we adopt the defaulted train/test data split. Regarding the features, we either use the feature files provided by the benchmark organizers or extract them by ourselves. They may not necessarily bring state-of-the-art accuracy since our focus is investigating the convergence speed of the optimization methods instead of just driving for higher performance. The defaulted tasks defined on some benchmarks are multi-class classification. In these cases, a one-vs-rest scheme is applied to simplify the evaluations. We pick the category with the most training samples as the positive class and merge all rest categories as the negative class, converting it into a binary classification problem. Whenever the positive/negative data partitions are heavily unbalanced, we assign samples from positive/negative classes different weights such that the weight summarizations of the two classes are equal. More formally, let $\mathcal{Y}_+, \mathcal{Y}_-$ be the index sets of positive/negative classes respectively. The loss is calculated as

$$P(w) = \frac{1}{|\mathcal{Y}_+|} \sum_{i \in \mathcal{Y}_+} \psi_i(w) + \frac{1}{|\mathcal{Y}_-|} \sum_{i \in \mathcal{Y}_-} \psi_i(w). \tag{34}$$

In all experiments we stick to using the logistic loss function and Tikhonov regularization owing to their empirical popularity and non-linear property.

### 4.2 Baseline Algorithms

We make comparisons between the proposed S3GD and other four competitors, including

- **Mini-Batch Stochastic Gradient Descent (SGD):** it represents the standard stochastic gradient method. At each iteration, the SGD algorithm randomly draws $p$ samples from the training set according to weight distribution specified in Eqn. (34), calculate their respective stochastic gradient, and uniformly average these stochastic gradients.

- **Stratified SGD (SSGD) [27]:** This method aims to improve the standard mini-batch SGD using data clustering and stratified sampling. SSGD ensures that each iteration will draw at least one sample from each data cluster (stratum). The inclusion is to contrast different ways of using global information about data. For fair comparison we set the number of clusters to be $p$.

- **Stochastic Variance Reduction Gradient (SVRG):** This original idea work of SVRG is found in [10]. However, it does not handle non-smooth functions. In the comparison we adopt the extension proposed in [22]. Inheriting the two nested loops of SVRG, one of the key parameters in [22] is the maximal iteration number in the inner loop. The authors suggest this parameter shall be sufficiently large for achieving better loss bound. We fix this parameter to be always 50 in all experiments, which empirically provides a good balance between convergence speed and heavy complexity caused by exact gradient estimation.

- **Stochastic Control Variate (SCV) [21]:** This is another semi-stochastic gradient method that reports state-of-the-art speed and accuracies. The method relies on the utilization of data statistics such as low-order moments to define “control variate”. The authors rigorously prove the reduction in noisy gradient variance under mild assumptions. Note that for features in high dimension, the computation of data statistics can be its computational bottleneck.

### 4.3 Evaluation Settings

For all experiments, we fix the parameter $\lambda = 10^{-3}$ for the Tikhonov regularization. The maximal iteration parameter $k_{in}$ in the inner loop of SGD is fixed to be 20. Each mini-batch contains $p = 10$ random samples. For SGD, $m = 100$ anchors are generated on all datasets. We implemented all baseline algorithms and S3GD in optimized C++ programs.
The experiments are conducted on shared servers in an industrial research lab. Each of the servers is equipped with 48 CPU cores and 400GB physical memory. Five independent trials are performed for all algorithms and the averaged results are reported. The entire experiments take about one day on five servers.

There are two indices which are utterly crucial for evaluating a gradient based optimization scheme: the correlation (or variance) between (semi)stochastic gradient and exact gradient, and the maximal step size which ensures the stability of the optimization. Large step sizes are always favored in practice since they expect improved convergence speed. In the literature of stochastic gradient methods, both decayed and constant step sizes are widely adopted. We found that the decayed step sizes did not work well compared with the constant ones on most data. Therefore we focus on the results using the constant step sizes. In Figure 2 we plot the objective values in each iteration of the training stage on CIFAR10. For all baseline algorithms and our proposed S3GD, the convergence curves under four different constant step sizes $\eta = \{0.1, 1, 5, 10\}$ are recorded and plotted. Obviously SVRG and S3GD are two most stable algorithms even operating with large step size parameters. All other three algorithms drastically fluctuate when their current solutions approach the global optimum, even with the moderate parameter $\eta = 5$. This empirical investigation highlights the importance of choosing proper step size. To fairly compare different algorithms, we evaluate them under the parameter set $\eta \in \{0.1, 1, 5, 10\}$ and report the performance with the largest step size that satisfies the following stability condition:

$$
\eta^* = \max \eta \quad s.t. \quad F(w; \eta) \leq (1 + \epsilon) F(w^*),
$$

where $F(w^*)$ denotes the objective value at the global optimum $w^*$. $F(w; \eta)$ is the converged point using step size $\eta$. In the experiments we average the last 5,000 optimization iterations to obtain $F(w; \eta)$. $\epsilon$ is set to be 0.01 in all cases. The condition aims to abandon any step size parameter that drives the solution crazily bounce around the global optimum.

Most of prior works [10], [21] report the performance with respect to iteration counts. We here argue that the evaluation shall take the iteration complexity into account. Recall that Table 1 summarizes the time and space iteration complexities for all algorithms. Importantly, the complexities of SVRG and SCV are dominated by the exact gradient computation and class-specific covariance matrix estimation. Both of them are expected to take longer time for accomplishing each iteration. Figure 3 reports the time for performing 50 gradient descent iterations for all datasets and algorithms. The computing time is obtained by averaging all trials. It is observed that on most datasets, the standard SGD consumes the least time. SSGD and our proposed S3GD use slightly more time compared to SGD. The CPU time of SVRG and SCV are significantly larger. Specifically, SVRG is especially slow in comparison when facing large scale data set (such as Kaggle-face and covtype) and high feature dimensions (e.g., 5,000-dimensional features for MED11 and 26,214-dimensional features for 20newsroups). Likewise, SCV is particularly slow when handling high-dimensional features. On the 20newsroups, SCV requires 594 seconds for every 50 iterations, which is beyond the scope of most practitioners. In contrast, SGD and S3GD only need 0.21 and 0.30 seconds respectively. Therefore for fairness in comparison, we will majorly concern the performance with respect to CPU times.

### 4.4 Quantitative Investigations

**Convergence Speed:** Figure 4 shows the training objective values with respect to the CPU times. For all algorithms, the step-size parameters are chosen according to the criterion in (35). Interestingly, though semi-stochastic gradient methods are proved to enjoy faster asymptotical convergence speed, most of them are not as “economic” as standard SGD due to significantly higher iteration complexity. Our proposed S3GD exceptionally outperforms all other algorithms on 6 out of 9 datasets. SVRG only dominates the small-scale 22-dimensional dataset IJCNN, and SGD yields the best performance on other two datasets KDD04 Bio and 20newsroups. SSGD is found to be sensitive about imbalanced data partition, such as MED11 and 20newsroups, where the positive/negative data ratios are 1:25 and 1:20 respectively.

It is surprising that the standard SGD is among the best performers on nearly all of 9 datasets despite its simplicity. Based on the experiments we argue that the research of semi-stochastic algorithms shall investigate the balance of larger step size and increased iteration complexity, particularly in the era of large data and high dimension.
Correlation of Gradients: We further study the Pearson correlation of (semi)stochastic gradient and the exact gradient. For a semi-stochastic algorithm, the correlation score is favored to approach the value of 1, since it indicates a better approximation scheme for gradient computation. It is clearly observed that SVRG and the proposed S3GD exhibit the most favorable correlation scores. Moreover, most methods enjoy relatively larger correlation scores when the optimization just begins. The correlation scores gradually drop when the optimization proceeds. The reason may be that the exact (sub)gradient tends to zero around the optimum, which makes accurate gradient approximation more challenging. The only exception is SVRG. In all cases its correlation scores quickly rise and stay at 1. It may be caused by the fact that \( \|w^{k+1} - w^k\| \) tends to zero when approaching the global optimum. Therefore, \( w \approx w^k \) in Eqn. (4), which implies that the semi-stochastic gradient becomes increasingly close to the full gradient.

Effect of Anchor: Recall that we use 100 anchors obtained through clustering in all experiments. One may concern how different choices of anchor number affect the performance and running time. Figure 6 presents the evolution of correlation scores under different anchor settings on MED11 and CIFAR10 (step size is fixed to be 1 for all cases). Interestingly, we observe that enlarging anchor set does not entail boosted correlation scores. In fact, the scores will reach its peak around data-specific anchor number (100 for MED11 and 20 for CIFAR10) though other choices bring alike performances. This implies that the algorithm is largely robust to the anchor number though empirical tuning does further help. Figure 7 plots the averaged iteration times for different anchor settings on MED11 and CIFAR10 (step size is fixed to be 1 for all cases). Interestingly, we observe that enlarging anchor set does not entail boosted correlation scores. In fact, the scores will reach its peak around data-specific anchor number (100 for MED11 and 20 for CIFAR10) though other choices bring alike performances. This implies that the algorithm is largely robust to the anchor number though empirical tuning does further help.

5 Concluding Remarks
In this paper we addressed the scalability issue pertaining to semi-stochastic gradient descent methods by proposing a novel approach S3GD. The motivation of S3GD is to reduce the high iteration complexity in existing semi-stochastic algorithms. It exploits stratified manifold-based gradient approximation as a good cure for the time-consuming exact gradient calculation. Our work significantly advances the original idea of residual-minimizing gradient correction.
The current paper did not discuss the application in a distributed computing environment, since it is out of the main scope. However, we will explore the distributed variants of the proposed S3GD like [19] in the future. Moreover, extension to non-convex formulations such as deep networks [7] is also a meaningful future direction.

Fig. 5. Pearson correlation scores of (semi)stochastic gradient and the exact gradient on 9 datasets under the parameter $\eta = 0.1$. See text for more explanation. Best viewing in color.

Fig. 6. Investigation of how the anchors affect gradient correlation scores on MED11 and CIFAR10.

Fig. 7. Average CPU time for every 50 optimization iterations (in seconds) for S3GD.

REFERENCES

[1] H. Bauschke, R. Goebel, Y. Lucet, and X. Wang. The proximal average: Basic theory. SIAM Journal on Optimization, 19(2):766–785, 2008.
[2] L. Bottou. Stochastic learning. In O. Bousquet and U. von Luxburg, editors, Advanced Lectures on Machine Learning. Springer Verlag, 2004.
[3] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. SIAM J. on Optimization, 20(4):1956–1982, 2010.
[4] O. Chapelle. Training a support vector machine in the primal. Neural Comput., 19(5):1155–1178, 2007.
[5] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. Better mini-batch algorithms via accelerated gradient methods. In NIPS, 2011.
[6] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. CoRR, abs/1012.1367, 2010.


Proof of Lemma 3.2:

Proof. Let us randomly choose \( w_0 \in \text{dom}(P) \) and introduce the auxiliary function as below

\[
q(w) = P(w) - \langle P'(w_0) - \mu_P w_0, w \rangle - \frac{\mu_P}{2} \|w\|^2. \tag{36}
\]

Note that \( q(w) \) remains a convex function since its Hessian matrix is still positive semi-definite. Moreover, it is easily verified that \( q(w) \) has the 0-strongly convexity and its Lipschitz parameters becomes \( L = L_P - \mu_P \). Taking the derivative with respect to \( w \) and set it zero, we have the optimum \( w^* = w_0 \). According to the definition of Lipschitz continuity, we have

\[
q(w^*) \leq q(w) - \frac{1}{L} q'(w) \leq q(w) - \frac{1}{L} \|q'(w)\|^2. \tag{37}
\]

Plugging in \( q'(w) = P'(w) - P'(w_0) + \mu_P w_0 - \mu_P w \) and applying the similar trick in the proof of Lemma 3.1, we have

\[
\langle q'(u) - q'(v), u - v \rangle \geq \frac{1}{L} \|q'(u) - q'(v)\|^2 \tag{38}
\]

Rearranging above inequality accomplishes the proof. \( \square \)

Combining above inequalities and applying Cauchy-Schwarz inequality will obtain the desired result. \( \square \)

APPENDIX

Proof of Lemma 3.1:

Proof. Let \( u' = \text{prox}_{\eta_R}(u) \) and \( v' = \text{prox}_{\eta_R}(v) \). Let \( \tilde{R}(u) = R(u) - (\mu_R/2)\|u\|^2 \). The assumption indicates that \( \tilde{R}(u) \) is also convex. From the optimality conditions we have,

\[
\frac{1}{\eta} \cdot (u - u' - \eta_R u') \in \partial \tilde{R}(u'),
\]

\[
\frac{1}{\eta} \cdot (v - v' - \eta_R v') \in \partial \tilde{R}(v').
\]

which indicate

\[
\tilde{R}(v') \geq \tilde{R}(u') + \frac{1}{\eta} (u - u' - \eta_R u')^\top (v' - u'),
\]

\[
\tilde{R}(u') \geq \tilde{R}(v') + \frac{1}{\eta} (v - v' - \eta_R v')^\top (u' - v'),
\]