Analysis of S2GD and its Boosted Variants

Varun Gupta
Chemical Engineering
Indian Institute of Technology, Kanpur

Manan Agarwal
Electrical Engineering
Indian Institute of Technology, Kanpur

Project Mode - Mixed

***Abstract*—Gradient Descent is one of the most widely used technique in the field of Machine Learning. This paper depicts the analysis of Semi-Stochastic Gradient Descent which includes convergence and evaluation of optimal parameters. We will further numerically compare it to the other variants of S2GD.**

I. Introduction

We analyse Semi-Stochastic Gradient Descent (S2GD) for minimization problem involving strongly convex functions. We critically observe the convergence of S2GD in Section II assuming general minimization problem of the form

$$\min_x f(x)$$

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

We assume that the convex functions are L-smooth and μ-strongly convex. While calculating the optimal parameters for S2GD we present a method to improve m(j), which is the maximum stochastic steps for $\nu = \mu$ case, and simulate it for various constraints. We numerically compare S2GD with other variants of GD for L2-regularized logistic regression in Section III. We also numerically analyse Loopless S2GD with S2GD and S2GD+ on a regression data-set. [CODE]

II. S2GD

A. Motivation

Both GD and SGD are extremely powerful but have their own disadvantages. GD needs large computation of full gradient whereas SGD gives a low accuracy solution. To improve fluctuation of SGD around optimum point we use full gradient.

B. Algorithm

Since S2GD comprises of both GD and SGD we will go over the iteration steps performed in the respective algorithms. Gradient Descent -

$$x_j = x_{j-1} - h\nabla f(x_{j-1})$$

Stochastic Gradient Descent

$$x_j = x_{j-1} - h\nabla f_i(x_{j-1})$$

The update rule for S2GD (for every epoch)

- Calculation of full gradient $g_j \leftarrow \nabla f(x_j)$ equivalent to n stochastic gradients
- Evaluating number of stochastic gradients (t_j) from geometric probability law
$$\frac{(1-\nu h)^{m-t}}{\beta} \quad and \quad \beta = \sum_{t=1}^{m}(1-\nu h)^{m-t}$$
h denotes the step size and m denotes the maximum stochastic steps for each epoch
- We initialize $y_{j,0} \leftarrow x_j$. For every step t = 1 to t_j we evaluate two stochastic gradients with update step being
$$y_{j,t} \leftarrow y_{j,t-1} - h(g_j + \nabla f_i(y_{j,t-1}) - \nabla f_i(x_j))$$
The direction vector in expectation gives the full gradient for the previous point $y_{j,t-1}$. This makes the update similar to an SGD update but in a non-standard way.
$$\mathbf{E}(g_j + \nabla f_i(y_{j,t-1}) - \nabla f_i(x_j)) = \nabla f(y_{j,t-1})$$
On completion of t_j steps we would store $x_{j+1} \leftarrow y_{j,t_j}$

C. Convergence Analysis

We assume that $f(x)$ is μ-strongly convex and $f_i(x)$ are L-smooth functions. We first define the sigma algebra ($\mathcal{F}_{j,t}$) for the problem comprising of the known variables $x_1, x_2, .., x_j$ and $y_{j,1}$,$y_{j,2}$,..,$y_{j,t}$.
We will begin analysing the convergence starting with expected distance from the current to the optimal point.

$$\mathbf{E}(\|y_{j,t} - x_*\|^2 \mid \mathcal{F}_{j,t-1}) = \|y_{j,t-1} - x_*\|^2 - 2h\langle y_{j,t-1} - x_*, \mathbf{E}(G_{j,t-1}|\mathcal{F}_{j,t-1})\rangle + h^2\mathbf{E}(\|G_{j,t-1}\|^2|\mathcal{F}_{j,t-1})$$

We define $G_{j,t-1}$ as the direction vector $g_j + \nabla f_i(y_{j,t-1}) - \nabla f_i(x_j)$. In the above equation we have written $y_{j,t}$ from the update rule. We begin by analysing $\mathbf{E}(\|G_{j,t-1}\|^2|\mathcal{F}_{j,t-1})$. We first state a useful property used in the derivation.

Lemma 1: *Given that a differentiable function is μ-strongly convex, following holds*

$$\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f(x_*))$$

Proof: We need to minimize the strong convexity inequality on both the sides [5]

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|^2$$

Minimizing LHS gives us f(x_*) whereas minimizing RHS with help of differentiation wrt y gives us

$$y = x - \frac{1}{\mu}\nabla f(x)$$

Plugging back the value for y we get the above property.

Theorem 2: *Given that a function f_i is L-smooth, the following inequality holds true*

$$\|\nabla f_i(x) - \nabla f_i(x_*)\|^2 \leq 2L[f_i(x) - f_i(x_*) - \langle \nabla f(x_*), x - x_* \rangle]$$

Taking expectation in Theorem 2 and $\nabla f(x_*) = 0$

$$\mathbf{E}(\|\nabla f_i(x) - \nabla f_i(x_*)\|^2) \leq 2L(f(x) - f(x_*))$$

Evaluating the expression for $\mathbf{E}(\|G_{j,t-1}\|^2 \mid \mathcal{F}_{j,t-1})$, expanding the expression and using the fact $\|x+y\|^2 \leq 2\|x\|^2 + 2\|y\|^2$ easily obtained using Cauchy-Schwartz and AM-GM.

$$\mathbf{E}(\|G_{j,t-1}\|^2 \mid \mathcal{F}_{j,t-1}) \leq 2\mathbf{E}(\|\nabla f_i(y_{j,t-1}) - \nabla f_i(x_*)\|^2) + 2\mathbf{E}(\|\nabla f_i(x_j) - \nabla f_i(x_*)\|^2) - 2\|\nabla f(x_j)\|^2$$

Using Both *Lemma 1* and *Theorem 2* in the above expression

$$\mathbf{E}(\|G_{j,t-1}\|^2 \mid \mathcal{F}_{j,t-1}) \leq 4L[f(y_{j,t-1} - f(x_*)] + 4(L-\mu)[f(x_j) - f(x_*)]$$

We have figured out the expected distance equation inform of known parameters. Substituting the above expression back

$$\mathbf{E}(\|y_{j,t} - x_*\|^2 \mid \mathcal{F}_{j,t-1}) \leq \|y_{j,t-1} - x_*\|^2 - 2h\langle y_{j,t-1} - x_*, \nabla f(y_{j,t-1})\rangle + 4Lh^2[f(y_{j,t-1}) - f(x_*)] + 4(L-\mu)h^2[f(x_j) - f(x_*)]$$

We will use the strong convexity inequality to evaluate the inner product in the above equation.

$$\nabla\langle f(y_{j,t-1}), (y_{j,t-1} - x_*)\rangle \geq f(y_{j,t-1}) - f(x_*) + \frac{\nu}{2}\|x_* - y_{j,t-1}\|^2$$

Since inner product has a negative sign substituting a lower value would not disturb the above inequality. Taking the expectation on both the sides.

$$\mathbf{E}(\|y_{j,t} - x_*\|^2) + 2h(1 - 2Lh)\mathbf{E}[f(y_{j,t-1}) - f(x_*)] \leq (1 - \nu h)\ \mathbf{E}(\|y_{j,t-1} - x_*\|^2) + 4(L-\mu)h^2\mathbf{E}[f(x_j) - f(x_*)]$$

We will be taking summation of the terms over both the sides by multiplying it with the geometric probability law and comparing the first terms on both side.

$$RHS \leftarrow \sum_{t=1}^{m}(1-\nu h)^{m-t}\mathbf{E}(\|y_{j,t} - x_*\|^2) = (1-\nu h)^{m-1} \mathbf{E}(\|y_{j,1} - x_*\|^2) + (1-\nu h)^{m-2}\mathbf{E}(\|y_{j,2} - x_*\|^2) ... + (1-\nu h)\mathbf{E}(\|y_{j,m-1} - x_*\|^2) + \mathbf{E}(\|y_{j,m} - x_*\|^2)$$

$$LHS \leftarrow \sum_{t=1}^{m}(1-\nu h)^{m-t+1}\mathbf{E}(\|y_{j,t-1} - x_*\|^2) = (1-\nu h)^{m}\mathbf{E}(\|x_j - x_*\|^2) + (1-\nu h)^{m-1}\mathbf{E}(\|y_{j,1} - x_*\|^2) ... + (1-\nu h)\mathbf{E}(\|y_{j,m-1} - x_*\|^2)$$

We can clearly observe middle terms getting cancelled on both RHS and LHS. Using the strong convexity inequality, thereby simplifying the above equation we get

$$\mathbf{E}(\|y_{j,m} - x_*\|^2) + 2\beta h(1-2Lh)\mathbf{E}(f(x_{j+1}) - f(x_*)) \leq 2\Big(\frac{(1-\nu h)^m}{\mu} + 2\beta(L-\mu)h^2\Big)\mathbf{E}(f(x_j) - f(x_*))$$

We thereby get the linear convergence for the algorithm assuming that $h \leq \frac{1}{2L}$ and hence ignoring the norm

$$\mathbf{E}(f(x_{j+1}) - f(x_*)) \leq c\mathbf{E}(f(x_j) - f(x_*))$$

$$c = \frac{(1-\nu h)^m}{\beta\mu h(1-2Lh)} + \frac{2(L-\mu)h}{1-2Lh}$$

While analysing the above expression we will be referring c = $c_1 + c_2$ respectively. Using the above inequality iteratively we can get

$$\mathbf{E}(f(x_{j+1}) - f(x_*)) \leq c^j\mathbf{E}(f(x_0) - f(x_*))$$

We now need to show that c must be less than 1 for the inequality to converge. We notice that as m $\to \infty$, $c_1 \to 0$ since $(1-\nu h)^m \to 0$ and $\beta \to \frac{1}{\nu h}$. Therefore we need $c_2 < 1$ which gives us the condition that

$$h < \frac{1}{4L - 2\mu}$$

Therefore for any positive value of h satisfying above inequality, there exist some value of m for which $c < 1$. We will now calculate the optimum parameters to minimize the computations. Obtaining a linear convergence paves a path for establishing a high probability result using Markov inequality.

$$\mathbf{P}\Big(\frac{f(x_j) - f(x_*)}{f(x_0) - f(x_*)} < \epsilon\Big) \geq 1 - \rho$$

where $\rho < 1$ and the above hold when

$$j \geq \frac{\log \epsilon\rho}{\log c}$$

D. Optimum Parameters

We express evaluating optimum parameter as a minimization problem over the total computational work. We assume that parameters related to problem are known such as L, μ, κ.

$$\min_{j,m,h} W = j(n + 2m)$$

The above relation is a simplification where we have substituted the upper bound on the number of stochastic gradients evaluated to simplify the total work done. The above constraint is subject to the linear convergence equation where $c^j < \epsilon$ which is the accuracy. We will define $\Delta = \epsilon^{1/j}$. We will analyse parameters for two different cases $\nu = 0$ and $\nu = \mu$.

- $\nu = 0$: We can obtain a closed form solution.

$$c = \frac{1}{m\mu h(1-2Lh)} + \frac{2(L-\mu)h}{1-2Lh} < \Delta$$

We will rearrange the above equation

$$m > \frac{1}{(\Delta(1-2Lh) - 2(L-\mu)h)}\frac{1}{\mu h}$$

We would minimize m by maximizing the quadratic equation in the denominator.

$$h = \frac{1}{\frac{4(L-\mu)}{\Delta} + 4L}$$

Substituting h in $c < \Delta$

$$m(j) > \frac{8(\kappa - 1)}{\Delta^2} + \frac{8\kappa}{\Delta}$$

- $\nu = \mu$: The paper follows the derivation that $c_2 = \Delta/2$ and $c_1 < \Delta/2$ to produce the parameters h(j) and m(j). We tried to generalize the above derivation to improve the coefficients for m(j). We substituted $c_2 = \Delta/x$ where x $>$ 1.

$$\frac{2(L-\mu)h}{1-2Lh} = \frac{\Delta}{x}$$

$$h = \frac{1}{\frac{2(L-\mu)x}{\Delta} + 2L}$$

We will evaluate $c_1 < \Delta - \Delta/x$

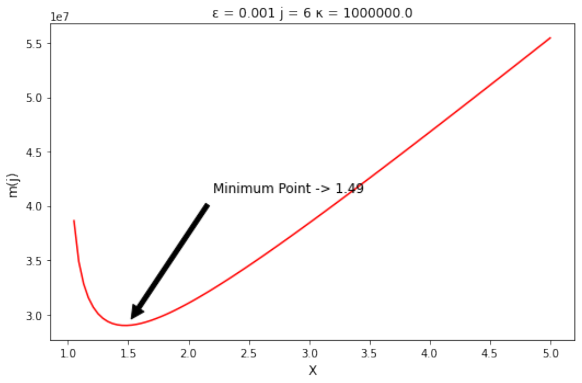
$$(1-\mu h)^m \left[1 + \frac{\Delta(x-1)}{x}(1-2Lh)\right] < \frac{\Delta(x-1)}{x}(1-2Lh)$$

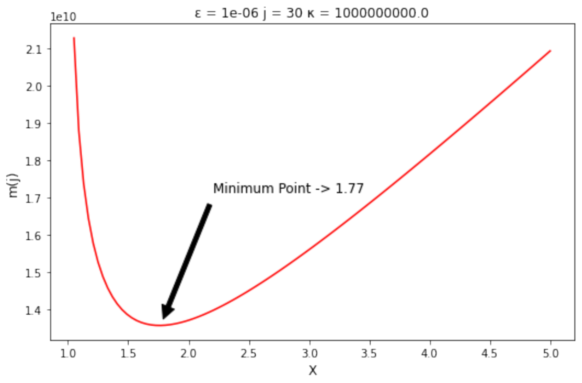
Taking *log* on both the sides

$$m \log\left[\frac{1}{1-\mu h}\right] > \log\left[\frac{1}{\frac{\Delta(x-1)(1-2Lh)}{x}} + 1\right]$$

Simplifying above and using the inequality $-\log(1 - \mu h) \geq \mu h$ we get the expression

$$m \geq \left(\frac{2x(\kappa-1)}{\Delta} + 2\kappa\right) \log\left[\frac{x}{(x-1)\Delta} + \frac{\frac{x\kappa}{x-1} - 1}{\kappa - 1}\right]$$

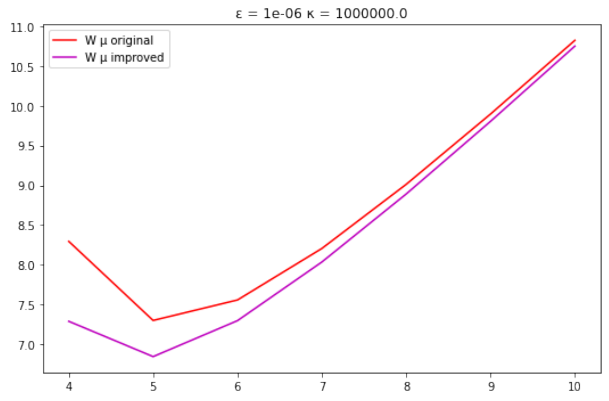




We will numerically simulate the improved result for different problem setups as shown above. General trend observed for the optimum point are -

- Decrease in ϵ, decrease in optimal point
- Increase in j, increase in optimal point
- No change observed in optimal point with change in κ

We simulated the total work in terms of full gradient evaluation and compared the result for a specific case described in the paper to show the improvement.



This gives us an idea that for different settings it is possible to obtain lesser value of computational work (W). We will now examine this hypothesis for the values presented in the paper.

		Previous Results		**Improved Results**	
ϵ	κ	j	W_μ	j	W_μ
1E-3	1E3	1	1.06n	1	1.04n
1E-6	1E3	2	2.12n	2	2.08n
1E-9	1E3	3	3.18n	3	3.12n
1E-3	1E6	3	3.78n	2	3.64n
1E-6	1E6	5	7.3n	5	6.84n
1E-9	1E6	8	10.9n	7	10.24n
1E-3	1E9	8	358n	8	344n
1E-6	1E9	16	717n	15	688n
1E-9	1E9	24	1076n	22	1032n

We have taken n = $1E9$ similar to that in the paper. We ran simulation for each problem for various values of j and mentioned the point where W was minimum. We can clearly observe that the total work done (W) is always less than the results obtained in the paper [1] even with lesser number of epochs for some of the cases. We have not considered $W(\nu = 0)$ case because we have obtained closed form solution in that case.

If we take j^* = O(log $\frac{1}{\epsilon}$), the complexity becomes of the form O((n+κ)log($\frac{1}{\epsilon}$)) which is asymptotically same as variance-reduced methods, SVRG and SDCA.

E. Convergence for convex loss

Now, we assume that $f(x)$ is a convex function and $f_i(x)$ are L-smooth functions. We define a new objective,

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\hat{f}_i(x) = f(x) + \frac{\mu}{2}||x - x_0||^2$$
$$\hat{f}_i(x) = f_i(x) + \frac{\mu}{2}||x - x_0||^2$$

$\hat{f}(x)$ is μ-strongly convex as $\hat{f}(x) - \frac{\mu}{2}||x||^2$ is a convex function. For lipschitz constant,

$$\begin{aligned}||\nabla\hat{f}_i(x) - \nabla\hat{f}_i(y)|| &= ||\nabla f_i(x) - \nabla f(y) + \mu(x - y)|| \\ &\leq (L + \mu)||x - y||\end{aligned}$$

Therefore, $\hat{f}_i$ are $L+\mu$ smooth. We take $\hat{x}_*$ to be the optimum of $\hat{f}(x)$. Now, we show an important lemma.

Lemma 3: *If* $\hat{x} \in R^D$ *satisfies,* $\hat{f}(\hat{x}) \leq \hat{f}(\hat{x}_*) + \delta$, $\delta > 0$, Then

$$f(\hat{x}) \leq f(x_*) + \delta + \frac{\mu}{2}||x_* - x_0||^2$$

Proof: We have,

$$\hat{f}(\hat{x}_*) = \min_{x\in R^D} f(x) + \frac{\mu}{2}||x - x_0||^2 \leq f(x_*) + \frac{\mu}{2}||x_* - x_0||^2$$

Adding δ on both sides and using the assumption,

$$\begin{aligned}&\hat{f}(\hat{x}) \leq f(x_*) + \delta + \frac{\mu}{2}||x_* - x_0||^2 \\ \implies &f(\hat{x}) \leq f(x_*) + \delta + \frac{\mu}{2}||x_* - x_0||^2\end{aligned}$$

using that $f \leq \hat{f}$.

Now we run the S2GD algorithm on $\hat{f}(x)$ by picking a random $x_0 \in R^D$. Let the iterates be $\hat{x}_0, .., \hat{x}_j$ where we take $\hat{x}_0 = x_0$. Using Lemma 2 and fact that if $A \implies B$ then $\mathbf{P}(A) \leq \mathbf{P}(B)$,

$$\begin{aligned}&\mathbf{P}\left(f(\hat{x}_j) - f(x_*) \leq \delta + \frac{\mu}{2}||x_* - x_0||^2\right) \\ &\geq \mathbf{P}\left(\hat{f}(\hat{x}_j) - \hat{f}(\hat{x}_*) \leq \delta\right)\end{aligned}$$

If $\delta = \epsilon(f(x_0) - f(x_*))$,

$$\mathbf{P}\left(\hat{f}(\hat{x}_j) - \hat{f}(\hat{x}_*) \leq \delta\right) \geq \mathbf{P}\left(\frac{\hat{f}(\hat{x}_j) - \hat{f}(\hat{x}_*)}{\hat{f}(\hat{x}_0) - \hat{f}(\hat{x}_*)} \leq \epsilon\right) \geq 1 - \rho$$

using the fact, $\hat{f}(\hat{x}_0) - \hat{f}(\hat{x}_*) \leq f(x_0) - f(x_*)$. If we now take $\mu = \epsilon$ and ϵ to be a small value, then difference of loss for j^{th} iterate i.e. $f(\hat{x}_j) - f(x_*)$ less than $K\epsilon$ holds with a high probability $1 - \rho$, if

$$j \geq \frac{\log \epsilon\rho}{\log \hat{c}}$$
$$\hat{c} = \frac{(1 - \nu h)^m}{\beta\mu h(1 - 2(L + \mu)h)} + \frac{2Lh}{1 - 2(L + \mu)h}$$

If $j_* = \log(\frac{1}{\epsilon})$ and $\mu = \epsilon$ the complexity of algorithm becomes $O\left((n + \frac{L}{\epsilon})\log(\frac{1}{\epsilon})\right)$

III. Numerical Simulation

We numerically simulate S2GD along with different variations of gradient methods to compare their performance. The github link is provided to load data-set inorder to run the codes locally Link

A. Binary Classification

We consider the L-2 regularised logisitic regression objective for binary classification problem. The training points are $\{\boldsymbol{x_n}, y_n\}_{n=1}^{N}$, where $\boldsymbol{x_n} \in R^D$ and $y_n \in \{-1, 1\}$. The likelihood is $p(y_n|\boldsymbol{x_n}, \boldsymbol{w}) = \sigma(y_n\boldsymbol{w}^T\boldsymbol{x_n})$, hence the overall objective is,

$$f(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N} f_n(\boldsymbol{w})$$
$$f_n(\boldsymbol{w}) = \log(1 + e^{-y_n\boldsymbol{w}^T\boldsymbol{x_n}}) + \frac{\lambda}{2}||\boldsymbol{w}||_2^2$$

Since the first part in $f_i(\boldsymbol{w})$ is convex the regularization term makes the objective $f(\boldsymbol{w})$, $\boldsymbol{\lambda}$ strongly convex. For a bound for the Lipschitz constant L, we use that for a differentiable function the maximum eigenvalue of its hessian provides an estimate for L. Computing the hessian,

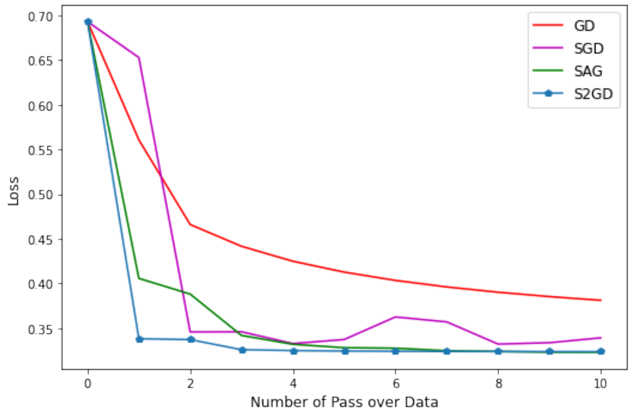
$$\nabla f(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}\frac{-y_n\boldsymbol{x_n}}{1 + e^{y_n\boldsymbol{w}^T\boldsymbol{x_n}}} + \lambda\boldsymbol{w}$$
$$\nabla^2 f(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}\frac{e^{y_n\boldsymbol{w}^T\boldsymbol{x_n}}}{(1 + e^{y_n\boldsymbol{w}^T\boldsymbol{x_n}})^2}.\boldsymbol{x_n}\boldsymbol{x_n}^T + \lambda\boldsymbol{I}$$

Hence,

$$\begin{aligned}L &= \sigma_{max}\left[\frac{1}{N}\sum_{n=1}^{N}\frac{e^{y_n\boldsymbol{w}^T\boldsymbol{x_n}}}{(1 + e^{y_n\boldsymbol{w}^T\boldsymbol{x_n}})^2}.\boldsymbol{x_n}\boldsymbol{x_n}^T\right] + \lambda \\ &\leq \frac{1}{4N}\sigma_{max}(\boldsymbol{X}^T\boldsymbol{X}) + \lambda\end{aligned}$$

Inequality follows as maximum value of $\frac{e^z}{(1+e^z)^2}$ is $\frac{1}{4}$ and $\boldsymbol{X}$ is $N \times D$ matrix of all training points.

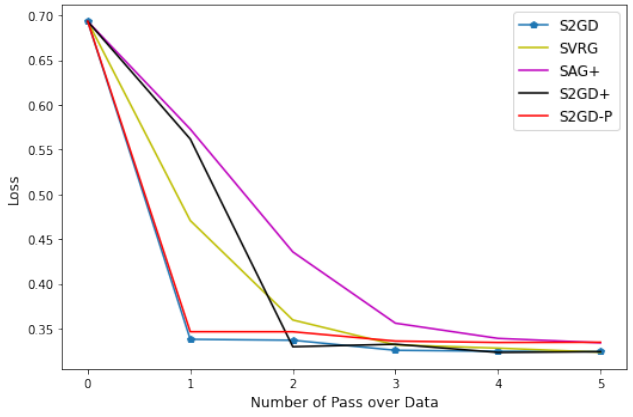
We will be using data-set *A9A* from *LIBSVM* library [6]. This data-set contains *32561* training points and *123* features.



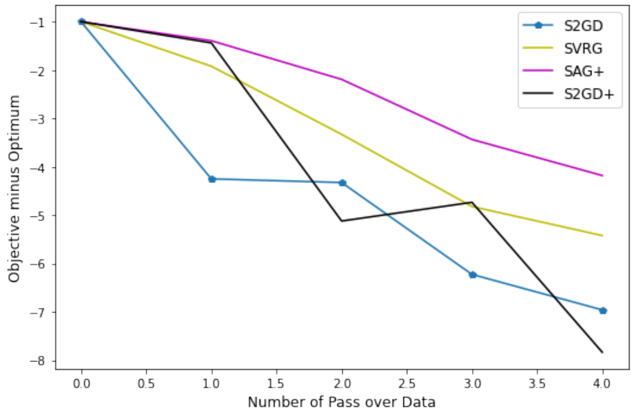
We will first simulate the data-set for number of passes over data to show the advantage of recently developed techniques over standard algorithms such as GD, SGD and SAG. We have taken the step size for GD as $\frac{1}{L}$ and SAG as $\frac{1}{16L}$. The paper [1] mentions the superior performance of S2GD+ over S2GD. The boosted version of S2GD+ as in [1]

- Run SGD for single pass over data (This improves the initial guess faster than GD)
- Run S2GD with fixed stochastic steps O(n)

We have also implemented S2GD-P, a new variant where we replaced the geometric distribution with Gaussian to analyse the specific use of certain distribution.



We tune the parameters for each algorithm separately. We observe that S2GD+ performs better than other boosted version of S2GD and SAG. We observe that S2GD-P is comparable to S2GD. We will now simulate $\log(f(x) - f(x_*)$ to observe the convergence rate.



B. *Multi-class Classification*

We consider the L-2 regularised softmax regression. We have K classes and learn weight vectors $\boldsymbol{w_1}, ..\boldsymbol{w_K}$. We learn them as a $D \times K$ matrix $\boldsymbol{W}$ which has the weight vectors as its columns. The likelihood is,

$$p(y_n = k|\boldsymbol{x_n}, \boldsymbol{W}) = \frac{e^{-\boldsymbol{w_k}^T x_n}}{\sum_{k=1}^{K} e^{-\boldsymbol{w_k}^T x_n}}$$

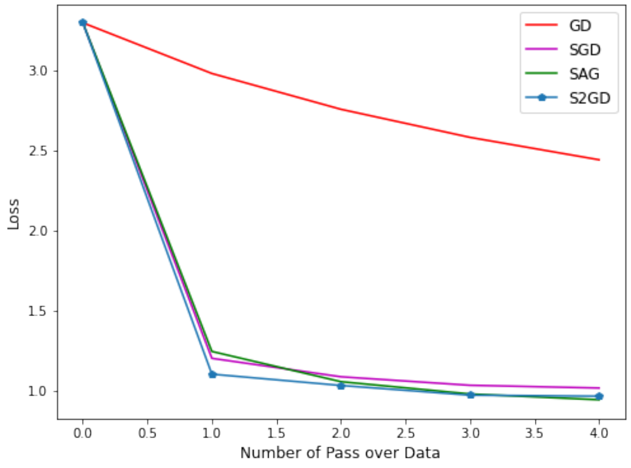
Consider $N \times D$ matrix $\boldsymbol{X}$ of all training points and $N \times K$ matrix $\boldsymbol{Y}$ of one hot encoding of all points. The overall objective is,

$$f(\boldsymbol{W}) = \frac{1}{N} \sum_{n=1}^{N} \left[\boldsymbol{w}_{y_n}{}^T x_n + \log(\sum_{k=1}^{K} e^{-\boldsymbol{w_k}^T \boldsymbol{x_n}}) \right] + \frac{\lambda}{2} ||\boldsymbol{W}||_2^2$$

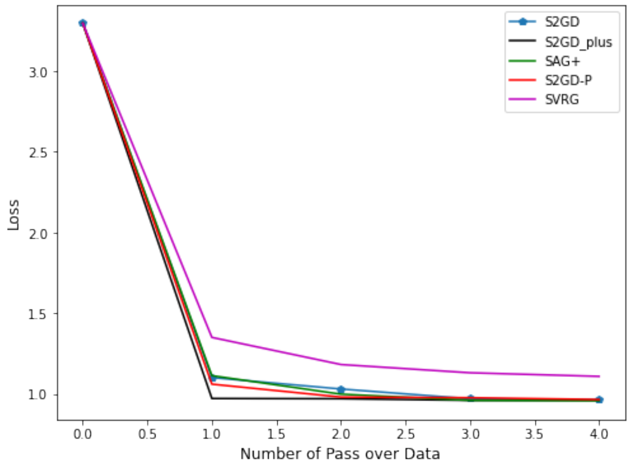
The gradient used is,

$$\nabla f(\boldsymbol{W}) = \frac{1}{N} (\boldsymbol{X}^T (\boldsymbol{Y} - \boldsymbol{S})) + \lambda \boldsymbol{W}$$

where $\boldsymbol{S}$ is a row-wise softmax matrix for $-\boldsymbol{XW}$. We use the above objective and gradient for numerical simulations. We use *letter* data-set with n = 15000, features = 16 and number of classes = 26 present in [6]. We will simulate the data in a similar manner as done previously.



We can clearly observe the disadvantage of GD with optimum step size over S2GD for same computational work.



After fine tuning, S2GD+ performs the best and we can also observe that S2GD-P performs slightly better than S2GD.

C. Linear Regression

We consider the L-2 regularised least squares problem. Consider $N \times D$ matrix $\boldsymbol{X}$ of all training points and $N \times 1$ vector $\boldsymbol{y}$ of all outputs. The objective is,

$$\begin{aligned} f(\boldsymbol{w}) &= \frac{1}{2N}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^T\boldsymbol{x_n})^2 + \frac{\lambda}{2}||\boldsymbol{w}||_2^2 \\ &= \frac{1}{2N}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||^2 + \frac{\lambda}{2}||\boldsymbol{w}||_2^2 \end{aligned}$$

Clearly $f(\boldsymbol{w})$ is λ-strongly convex. The gradient used and hessian are,

$$\begin{aligned} \nabla f(\boldsymbol{w}) &= \frac{1}{N}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) + \lambda\boldsymbol{w} \\ \nabla^2 f(\boldsymbol{w}) &= \frac{1}{N}\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I} \end{aligned}$$

Hence, $L = \frac{1}{N}\sigma_{max}(\boldsymbol{X}^T\boldsymbol{X}) + \lambda$.
For the above problem we simulate Loopless S2GD along with S2GD and S2GD+. The motivation for Loopless S2GD is taken from [7] where we skip the hard-thresholding step as we do not have a sparsity constraint. Algorithm follows,

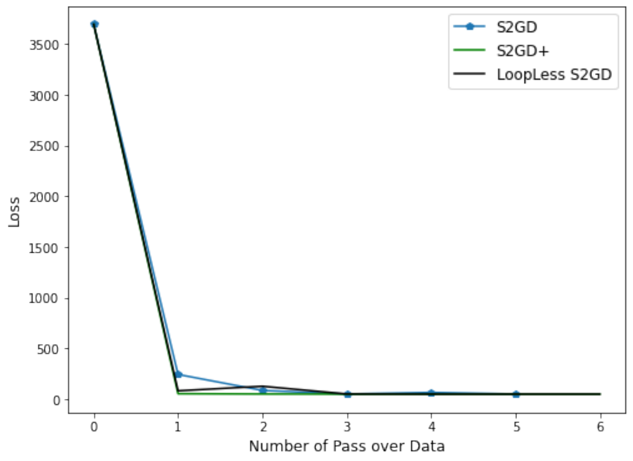
- For every iteration update,

$$x^{(t+1)} = x^{(t)} - \eta(\nabla f_i(x^t) - \nabla f_i(w^t) + \nabla f(w^t))$$

- Update w as,

$$\begin{aligned} w^{(t+1)} &= x^{(t)} && \text{(with probability p)} \\ w^{(t+1)} &= w^{(t)} && \text{(with probability 1-p)} \end{aligned}$$

We have used the data-set *cpusmall* from [6] which has *8192* training points with *12* features.



While the performance is comparable for all the three algorithms, we can observe that S2GD+ and Loopless S2GD show better initial convergence than S2GD. When compared to GD and SGD these algorithm outperform by a large margin showing better initial convergence with far less variance.

IV. Conclusion

In our paper, we have critically analysed the convergence of the S2GD algorithm for strongly convex functions. For the case of convex functions we have shown that we are able to approach the optimum loss in a very tight neighbourhood and this holds with sufficiently high probability. We have tabulated an updated lower bound for m(j) for different known constraints which results in a lesser computational work and can be referred to in future for setting better optimal parameters. We numerically simulated S2GD with different variants of gradient descent algorithms on different objectives to confirm its consistency in performance for empirical risk minimisation problem. We tried out S2GD-P which follows Gaussian distribution for selecting stochastic steps in inner loop and it performed comparable with Boosted version S2GD+. At last we simulated Loopless S2GD which omits outer loop and is still closely related to S2GD.

References

[1] Peter Richtarik and Jakub Konecny, “Semi-Stochastic Gradient Descent Methods” arXiv:1312.1666.
[2] Rie Johnson and Tong Zhang, “Accelerating Stochastic Gradient Descent using Predictive Variance Reduction”. in NIPS, 2013.
[3] Mark Schmidt, Nicolas Le Roux and Francis Bach, “Minimizing Finite Sums with the Stochastic Average Gradient” arXiv:1309.2388.
[4] Yurii Nesterov, “Introductory lectures on convex optimization: A basic course”, volume 87. Springer, 2004.
[5] Strong Convexity, Xingyu Zhou “https://xingyuzhou.org/blog/notes/strong-convexity”
[6] Datasets “https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets”
[7] Xiangyang Liu and Bingkun Wei, ”Loopless Semi-Stochastic Gradient Descent with Less Hard Thresholding for Sparse Learning”