IE510 Term Paper: Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz Algorithm

Tanmay Gupta
Department of Computer Science
University of Illinois at Urbana Champaign
tgupta6@illinois.edu

Aditya Deshpande
Department of Computer Science
University of Illinois at Urbana Champaign
ardeshp2@illinois.edu

Abstract

In this paper, we mainly study the convergence properties of stochastic gradient descent (SGD) as described in Needell et al. [2]. The function to be minimized with SGD is assumed to be strongly convex. Also, its gradients are assumed to be Lipschitz continuous. First, we discuss the superior bound on convergence (of standard SGD) obtained by Needell et al. [2] as opposed to the previous work of Bach and Moulines [1]. Then, we show that this bound can be further improved if SGD is performed with importance (weighted) sampling instead of uniform sampling. Finally, we study two applications: Logistic Regression and Kaczmarz Algorithm and demonstrate faster convergence obtained from SGD with weighted (or partially weighted) sampling.

1 Stochastic Gradient Descent

In stochastic gradient descent (SGD), we minimize a function $F(w)$ using stochastic gradients in the update rule at each step. The stochastic gradients $\mathbf{g}$ are such that their expectation is the gradient of $F(w)$, $\mathbb{E}[\mathbf{g}] = \nabla F(w)$. Now, if $F(w) = \mathbb{E}_{i\sim\mathcal{D}}[f_i(w)]$, then we have $g = \nabla f_i(w)$. The update rule then is as follows,

$$w_{k+1} = w_k - \gamma \nabla f_{i_k}(w), \tag{1}$$

where γ is the step size and i_k is drawn i.i.d from some distribution $\mathcal{D}$.

In this paper, we will study the convergence of SGD on the function F, with the following assumptions

1. Each f_i is continuously differentiable and the gradient of f_i has a Lipschitz constant L_i, i.e. $\|\nabla f_i(w_1) - \nabla f_i(w_2)\|_2 \leq L_i \|w_1 - w_2\|_2$
2. F is strongly convex with parameter μ, i.e.

   $F(w_1) \geq F(w_2) + \nabla F(w_2)^T (w_1 - w_2) + \frac{\mu}{2}\|w_1 - w_2\|_2^2$, or equivalently

   $(w_1 - w_2)^T (\nabla F(w_1) - \nabla F(w_2)) \geq \mu \|w_1 - w_2\|_2^2$

2 Convergence of SGD

Convergence up to a tolerance ϵ, is defined as obtaining a solution w_k such that $\mathbb{E}[\|w_k - w_*\|] \leq \epsilon$, where w_* is the optimal solution. In this section, we find minimum number of iterations, k, required by different methods so that the desired tolerance is guaranteed.

2.1 Bach and Moulines

Bach and Moulines [1], showed that

$$k = 2log(\frac{\epsilon_0}{\epsilon})(\frac{\mathbb{E}L_i^2}{\mu^2} + \frac{\sigma^2}{\mu^2\epsilon}) \tag{2}$$

iterations of eq (1) with an appropriate step-size are sufficient to ensure convergence to a desired tolerance ϵ. Here, $\sigma^2 = \mathbb{E}[\|\nabla f_i(w_*)\|_2^2]$. When w_* is the unique minimizer for all f_i, we have $\sigma^2 = 0$. In this case, the number of iterations scales logarithmically as $\frac{1}{\epsilon}$ (exponential convergence), with a further constant given by the average squared conditioning number $\mathbb{E}[(\frac{L_i}{\mu})^2]$. If $\sigma^2 \neq 0$, we have the additional term of $\frac{\sigma^2}{\mu^2\epsilon}$.

Note that, in eq (2) the number of iterations scales with the expected squared conditioning number $\mathbb{E}[(\frac{L}{\mu})^2]$. We call this the quadratic dependence. As shown by Needell et al. [2], this dependence is first reduced to $\frac{\sup L}{\mu}$ (Section 2.2) and then further to linear convergence, $\frac{L}{\mu}$ (Section 3.3).

2.2 Quadratic $\mathbb{E}[(\frac{L}{\mu})^2]$ to $\frac{supL}{\mu}$

In contrast to Bach and Moulines [1], Needell et al. use a tighter recurrence relation on the SGD iterates ([2], Section A.2) and obtain

$$\mathbb{E}[\|w_k - w_*\|] \leq (1 - 2\gamma\mu(1 - \gamma \sup L))^k \|w_0 - w_*\|^2 + \frac{\gamma\sigma^2}{\mu(1 - \gamma \sup L)} \tag{3}$$

where step-size $\gamma < \frac{1}{\sup L}$, and $\epsilon_0 = \|w_0 - w_*\|^2$.

Substituting $\gamma = \frac{\mu\epsilon}{2\epsilon\mu \sup L + 2\sigma^2}$ gives the second term in eq (3) is less than $\epsilon/2$. Now, requiring the first term $(1-2\gamma\mu(1-\gamma \sup L))^k\epsilon_0^2$ to be also less than $\epsilon/2$, we obtain a lower bound on the number of iterations

$$k \geq 2log(\frac{2\epsilon_0}{\epsilon})(\frac{\sup L}{\mu} + \frac{\sigma^2}{\mu^2\epsilon}) \tag{4}$$

to get the desired tolerance of ϵ. With step size $\gamma = \frac{\mu\epsilon}{2\epsilon\mu \sup L + 2\sigma^2}$ and k iterations (as per eq 4), SGD now converges in steps proportional to $\frac{\sup L}{\mu}$ instead of $\mathbb{E}[(\frac{L}{\mu})^2]$.

3 Weighted SGD

In this section we discuss the use of importance sampling to improve convergence properties of SGD. We will first introduce importance sampling and then show how it fits in context of SGD.

3.1 Importance Sampling

Importance sampling is a technique to compute expectations of a random variable $Y = f(X)$ with respect to a distribution $p(X)$ by drawing samples from a weighted distribution $q(x) \propto p(X)g(X)$ instead of the original distribution $p(X)$. Importance sampling is done either to reduce variance of

the estimate or if sampling from the weighted distribution is easier than sampling from the original distribution.

We can compute expectation of Y with respect to $p(X)$ as follows:

$$\mathbb{E}_p[Y] = \sum_x f(x)p(x) \tag{5}$$

$$= \sum_x \frac{f(x)}{g(x)} p(x)g(x) \tag{6}$$

$$= Z \sum_x \frac{f(x)}{g(x)} \frac{p(x)g(x)}{Z} \qquad \text{(where } Z = \textstyle\sum_x p(x)g(x)\text{)}$$

$$= Z \sum_x \frac{f(x)}{g(x)} q(x) \tag{7}$$

Now if we choose $g(x)$ such that $Z = 1$ then we have

$$\mathbb{E}_p[Y] = \sum_x \frac{f(x)}{g(x)} q(x) \tag{8}$$

This implies that we can compute the desired expectation simply by computing the expectation of the modified random variable $Y' = \frac{f(X)}{g(X)}$ with respect to the modified distribution $q(x) = \frac{f(x)g(x)}{Z}$.

3.2 Importance Sampling for SGD

Now we describe how importance sampling can be used for weighted SGD. We begin by noting that in SGD we minimize $F(w) = \mathbb{E}_{i\sim p(i)}[f_i(w)]$ by drawing a random sample $i_k \sim p(i)$ and using the update rule $w_{k+1} = w_k - \gamma \nabla f_{i_k}(w)$. Using importance sampling we know that

$$F(w) = \mathbb{E}_{i\sim p(i)}[f_i(w)] \tag{9}$$

$$= \mathbb{E}_{i\sim p(i)g(i)} \left[\frac{f_i(w)}{g(i)} \right] \qquad \text{(where g is chosen such that } \textstyle\sum_i p(i)g(i) = 1\text{)}$$

So now the weighted SGD can can be described by the following two steps:

1. Draw a sample: $i_k \sim p(i)g(i)$
2. Update the parameters: $w_{k+1} = w_k - \frac{\gamma}{g(i)} \nabla f_{i_k}(w)$

Now, to get the convergence bound we simply need to replace L_i and σ for the original function f_i by $L_i^{(g)}$ and $\sigma^{(g)}$ for the weighted function $f_i^{(g)} = \frac{f_i(w)}{g(i)}$ and choose g which minimizes $\sup L^{(g)}$.

3.2.1 Computing g and $L_i^{(g)}$

Since the modified function is simply a scalar $\frac{1}{g(i)}$ multiplied to the original function $f_i(w)$,

$$L_i^{(g)} = \frac{L_i}{g(i)} \tag{10}$$

Now, we note that our convergence bound depends upon $\sup L_i^{(g)} = \frac{L_i}{g(i)}$ which is minimized for

$$g(i) = \frac{L_i}{\mathbb{E}[L_i]} = \frac{L_i}{\overline{L}} \tag{11}$$

For this choice of g the Lipschitz constants for the updated functions can be computed as $L_i^{(g)} = \overline{L}$ from equation 10. Note that this also implies $\sup L = \overline{L}$.

3.2.2 Computing $\sigma^{(g)}$

We can compute a bound on $\sigma^{(g)}$ for $g(i) = \frac{L_i}{\overline{L}}$ as follows

$$(\sigma^{(g)})^2 = \mathbb{E}_{i\sim p(i)g(i)}\left[\|\nabla f_i^{(g)}(w^*)\|_2^2\right] \tag{12}$$

$$= \mathbb{E}_{i\sim p(i)g(i)}\left[\frac{1}{g(i)^2}\|\nabla f_i(w^*)\|_2^2\right] \tag{13}$$

$$= \mathbb{E}_{i\sim p(i)}\left[\frac{1}{g(i)}\|\nabla f_i(w^*)\|_2^2\right] \tag{14}$$

$$= \mathbb{E}_{i\sim p(i)}\left[\frac{\overline{L}}{L_i}\|\nabla f_i(w^*)\|_2^2\right] \tag{15}$$

$$\leq \frac{\overline{L}}{\inf L}\sigma^2 \tag{16}$$

3.3 Improved convergence with Weighted SGD

Using the above computed values in equation 4 we get a new bound which depends linearly on $\overline{L}$ instead of $\sup L$.

$$k \leq 2\log\left(\frac{2\epsilon_0}{\epsilon}\right)\left(\frac{\overline{L}}{\mu} + \frac{\overline{L}}{\inf L}\cdot\frac{\sigma^2}{\mu^2\epsilon}\right) \tag{17}$$

However, we also note that this leads to an increased dependence on σ^2 by a factor of $\frac{\overline{L}}{\inf L}$. If the variance in L is large, $\frac{\overline{L}}{\inf L} >> 1$ and we see a larger dependence on σ^2.

3.4 Reducing dependence on σ^2

We can reduce the dependence on σ^2 by using a mixture of the source distribution and its re-weighting

$$g(i) = \frac{1}{2} + \frac{1}{2}\cdot\frac{L_i}{\overline{L}} \tag{18}$$

We now need to re-compute $\sup L_i^{(g)}$ and $\sigma^{(g)}$ for the above weights which are as follows:

1. Computation of $\sup L_i^{(g)}$ for new weights

$$\sup L_i^{(g)} = \sup\frac{L_i}{\frac{1}{2} + \frac{1}{2}\cdot\frac{L_i}{\overline{L}}} \tag{19}$$

$$\leq 2\overline{L} \tag{20}$$

2. Computation of $(\sigma^{(g)})^2$ for new weights

$$(\sigma^{(g)})^2 = \mathbb{E}_{i\sim p(i)}\left[\frac{1}{\frac{1}{2} + \frac{1}{2}\cdot\frac{L_i}{\overline{L}}}\|\nabla f_i(w^*)\|_2^2\right] \tag{21}$$

$$\leq 2\sigma^2 \tag{22}$$

With these newly computed values we get a new convergence bound using equation 4

$$k \leq 4\log\left(\frac{2\epsilon_0}{\epsilon}\right)\left(\frac{\overline{L}}{\mu} + \frac{\sigma^2}{\mu^2\epsilon}\right) \tag{23}$$

Note that these bounds hold true only for step sizes chosen according to section 2.2.

For the complete algorithm refer to Algorithm 3.1 Needell et al.[2].

4 Application

We now show application of weighted SGD for 2 practical optimization problems - L2 regularized Logistic regression and Linear Least Squares (Kaczmarz). Since the former is our main contribution in this term paper we derive the Lipschitz constants and strong convexity parameter for it in the subsequent subsections. Kaczmarz algorithm was presented in the original paper so we simply state its results.

4.1 Logistic Regression

Logistic regression is a popular machine learning algorithm used for binary classification using a linear model. We formally define the problem as follows:

- Given: A set of samples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^n$ and $y_i \in \{1, -1\}$
- Parameters: $w \in \mathbb{R}^n$
- Model: $P(y = y_i|x_i) = \frac{1}{1+e^{-y_i w^T x_i}}$

This can be cast as the following unconstrained optimization problem

$$\min_w F(w) = \frac{\|w\|^2}{2} + C\sum_{i=1}^{n} \log(1 + e^{-y_i w^T x_i}) \tag{24}$$

Note that $F(w) = \mathbb{E}_{i\sim Uniform(n)}\left[\frac{\|w\|^2}{2} + Cn\log(1 + e^{-y_i w^T x_i})\right]$ and hence fits into the SGD framework with $f_i(w) = \frac{\|w\|^2}{2} + Cn\log(1 + e^{-y_i w^T x_i})$ and $p(i) = \frac{1}{n}$. The gradient can be computed as follows

$$\begin{aligned}
\nabla f_i(w) &= w - Cny_i x_i \frac{e^{-y_i w^T x_i}}{1 + e^{-y_i w^T x_i}} && (25)\\
&= w - Cny_i x_i h(y_i w^T x_i) && (\text{where } h(t) = \tfrac{1}{1+e^t})
\end{aligned}$$

4.1.1 Finding Lipschitz Constant

The Lipschitz constants for the gradients can be found as follows:

$$\begin{aligned}
\|\nabla f_i(w_1) - \nabla f_i(w_2)\| &= \|(w_1 - w_2) - Cny_i x_i(h(y_i w_1^T x_i) - h(y_i w_2^T x_i))\| && (26)\\
&\leq \|w_1 - w_2\| + |Cn(h(y_i w_1^T x_i) - h(y_i w_2^T x_i))|\|x_i\| && (27)
\end{aligned}$$

Using mean value theorem on h we get

$$\begin{aligned}
h(t_1) - h(t_2) &= \nabla h(\xi)(t_1 - t_2) && (28)\\
&= -h(\xi)(1 - h(\xi))(t_1 - t_2) && (\text{For some } \xi \in [t_1, t_2])\\
&\geq -0.25(t_1 - t_2) && (\text{Since } h(t) \in [0, 1])\\
\implies h(y_i w_1^T x_i) - h(y_i w_2^T x_i)) &\geq -0.25(w_1 - w_2)^T x_i && (29)
\end{aligned}$$

Using equations 27 and 29 we get

$$\begin{aligned}
\|\nabla f_i(w_1) - \nabla f_i(w_2)\| &\leq (1 + 0.25Cn\|x_i\|^2)\|w_1 - w_2\| && (30)\\
\implies L_i &= 1 + 0.25Cn\|x_i\|^2 && (31)
\end{aligned}$$

4.1.2 Finding Convexity Parameter

The strong convexity parameter can be found as follows

$$(\nabla F(w_1) - \nabla F(w_2))^T (w_1 - w_2) \tag{32}$$

$$= \left[(w_1 - w_2) - C \sum_{i=1}^{n} y_i x_i (h(y_i w_1^T x_i) - h(y_i w_2^T x_i)) \right]^T (w_1 - w_2) \tag{33}$$

$$= \left[(w_1 - w_2) + C \sum_{i=1}^{n} x_i h(\xi)(1 - h(\xi))(w_1 - w_2)^T x_i \right]^T (w_1 - w_2)$$

(Using Mean Value Theorem)

$$= \|w_1 - w_2\|^2 + C \sum_{i=1}^{n} h(\xi)(1 - h(\xi))((w_1 - w_2)^T x_i)^2 \tag{34}$$

$$\geq \|w_1 - w_2\|^2 \tag{35}$$

$$\implies \mu = 1 \tag{36}$$

4.1.3 Weights for Importance Sampling

As discussed earlier in Section 3.2.1, we use weights

$$g(i) = \frac{L_i}{\overline{L}} = \frac{1 + 0.25Cn\|x_i\|^2}{\sum_i 1 + 0.25Cn\|x_i\|^2} \tag{37}$$

Note that, for this case, data points further away from the origin get high weights and are sampled frequently.

4.1.4 Convergence Properties

If we substitute 31 in 17, we obtain the bound on iterations for the convergence of weighted SGD. It can be seen that, a large variance in L_i (which is proportional to variance in $\|x_i\|^2$), results in a larger dependence on σ^2.

4.2 Kaczmarz Algorithm

Kaczmarz Algorithm involves solving a linear least squares problem $F(x)$ to find optimal x_*, where

$$F(x) = \frac{1}{2} \sum_{i=1}^{n} (a_i^T x - b_i)^2 = \frac{1}{2} \|Ax - b\|_2^2 \tag{38}$$

The update rule used is

$$x_{k+1} = x_k + c \cdot \frac{b_i - a_i^T x_k}{\|a_i\|} a_i \tag{39}$$

where i is sampled from a probability distribution with probability $g(i) = \frac{n\|a_i\|^2}{\|A\|^2}$ for every update.

4.2.1 Connection to Weighted SGD:

Needell et al. [2] show that this Kaczmarz Algorithm is a specific case of their general weighted SGD framework. Using $f_i = \frac{n}{2}(a_i^T x - b_i)^2$, we obtain $L_i = n\|a_i\|^2$ and $\mu = \frac{1}{\|(A^T A)^{-1}\|_2^2}$. One can see that substituting weights proportional to these L_i in weighted update rule of SGD will give the update rule of Kaczmarz again. Also, Kaczmarz samples i with probability $g(i)$, which is same as weights obtained using $\frac{L_i}{\overline{L}}$ of weighted SGD.

5 Experimental Results

5.1 Logistic Regression

5.1.1 Datasets

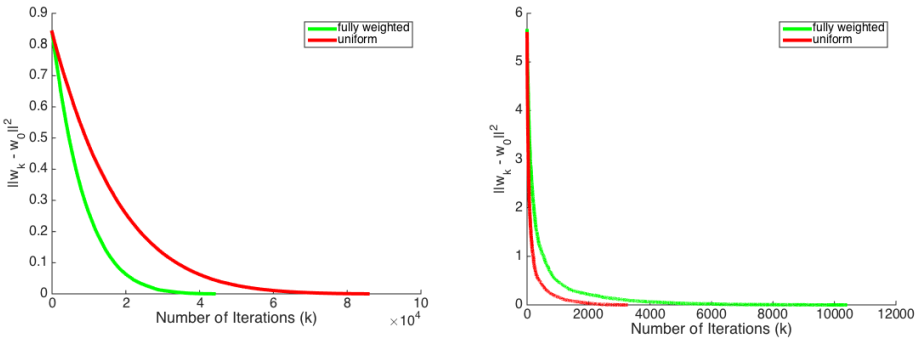
We generate two binary classification datasets: (I) and (II). The classes are sampled from two 2D gaussian distributions. The mean and the variance are chosen so that the classes are linearly separable. The separating hyperplane is roughly the x-axis. For (I), the means for the two guassian distributions are $(0, 15)$ and $(0, -15)$ and the variance is 5. For (II), we divide the data in (I) by 10, now the means are $(0, \pm 1.5)$ with variance $.5$. We perform a standard gradient descent to obtain the optimal solution w_* for our datasets.

5.1.2 Uniform SGD vs. Weighted SGD

In what follows, we compare the convergence of the uniform and weighted SGD algorithms on the two datasets.

Data (I): As shown in eq (37), our Lipschitz constants L_i are proportional to $\|x_i\|$. Note, for (I), $\|x_i\|$ is large because data is drawn from a distribution centered at $(0, \pm 15)$ with high variance. Thus, $\sup(L)$ is larger than $\mathbb{E}[L]$. As a result, we obtain better convergence with weighted SGD than uniform SGD (Figure 1).

Data (II): Since the variance and mean reduce by a factor of 10, $\|x_i\|$ and thus $\|L_i\|$ are now smaller. Also, due to low variance $\sup L$ and $\mathbb{E}[L]$ are similar. In Figure 1, now that $\sup L$ and $\mathbb{E}[L]$ are smaller than (I), we observe faster convergence of uniform and weighted SGD on (II). Note that uniform SGD converges faster than weighted SGD. We believe this is because the term depending on σ^2 dominates the number of iterations as opposed to small L_i term. Comparing equations (4) and (17), one can see that the σ^2 term is larger in weighted SGD and thus, it converges slower than uniform SGD.



(a) Dataset (I) (b) Dataset (II)

Figure 1: Convergence of uniform SGD and weighted SGD. On dataset (I), weighted SGD is faster than uniform SGD because of smaller $\mathbb{E}[L]$ than $\sup L$. And, weighted SGD is slower on dataset (II) since the larger σ^2 term in the number of iterations for weighted SGD now dominates.

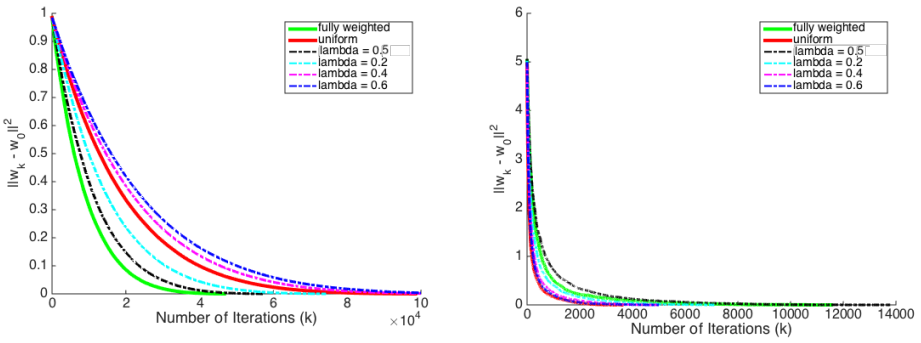
5.1.3 Benefits of Partial Weighted SGD

As shown in Section 3.4, we can use a combination of uniform and weighted distribution with a mixing co-efficient λ as follows:

$$g(i) = \lambda \cdot 1 + (1 - \lambda) \cdot \frac{L_i}{\overline{L}} \tag{40}$$

We call the SGD algorithm that uses the weights $g(i)$ of equation (40) with $\lambda \in (0, 1)$ as partially weighted SGD. Note, now $\lambda = 0$ corresponds to weighted SGD and $\lambda = 1$ corresponds to uniform

SGD. As seen in Figure 2, on dataset (II), where weighted SGD was slower, with $\lambda = .6$ we again obtain a faster convergence. Also, partially weighted SGD does not adversely affect dataset (I) and $\lambda = .5$ shows similar convergence as fully weighted SGD. The use of λ, allows the method to be robust to different types of datasets.



(a) Dataset (I) (b) Dataset (II)

Figure 2: Convergence of uniform, weighted and partially SGD for datasets (I) and (II). Partially weighted SGD is a trade-off between uniform and weighted SGD and for an appropriate λ gives fast convergence.

5.2 Kaczmarz Algorithm

In this section, we discuss the results obtained by Needell et al. [2] for their application of weighted SGD on the Kaczmarz Algorithm.

In Figure 3 Case 1, we have a single large L_i, thus $\sup L$ is much larger than $\mathbb{E}[L]$. As a result, uniform SGD ($\lambda = 1$) is much slower to converge than weighted ($\lambda = 0$). In Figure 3 Case 2, when there are no outlier L_i, $\sup L$ and $\mathbb{E}[L]$ are similar and both algorithms show similar convergence behavior.

In Figure 4 Case 3 to 5, we reduce the value of σ^2 from large to small. The σ^2 term is larger in weighted SGD as seen in equation 17 (because of $\frac{\overline{L}}{\inf L}$). Thus, weighted SGD converges very slowly in Case 3 and its convergence improves as σ^2 becomes small.

Again, these results show that we can be robust to different datasets (i.e. different values of L, σ) if we use partially weighted SGD. In all cases of Figure 3, 4, we see that there exists some λ that provides better performance than pure uniform or pure weighted sampling.

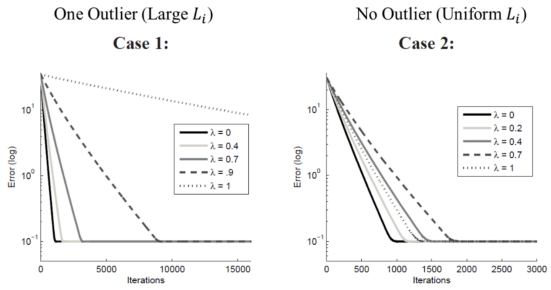


Figure 3: Performance of different SGD algorithms when L varies.

References

[1] F. Bach and E. Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. Advances in Neural Information Processing Systems (NIPS), 2011.

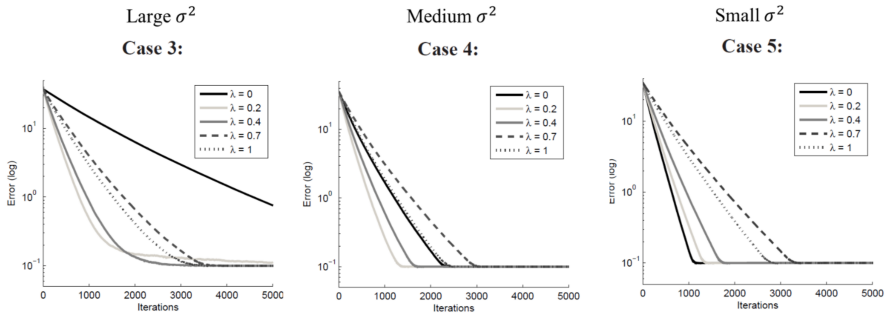


Figure 4: Performance of different SGD algorithms when σ^2 varies.

[2] D. Needell, N. Srebro and R. Ward. Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz algorithm. Proc. Neural Information Processing Systems (NIPS), Dec. 2014 (http://arxiv.org/pdf/1310.5715v5.pdf).