

CSED490Y: Optimization for Machine Learning

Week 12: Variance reduced methods

Namhoon Lee

POSTECH

Spring 2022

15mins / group

Midway group presentation:

- ▶ Schedule:
 - ▶ May 11: groups 6, 4, 3, 7, 10
 - ▶ May 18: groups 1, 5, 9, 11, 8
 - ▶ May 25: groups 2, 12, 13 — Quiz 2
- ▶ Upload your slides on PLMS by **11am of the presentation day.**

Recap: SGD

Stochastic oracle $x \rightarrow \boxed{\text{oracle}} \rightarrow \tilde{g}_x \quad \mathbb{E}[\tilde{g}_x] = \nabla f(x)$

Finite sum $\min_x f(x) := \frac{1}{n} \sum_{i=1}^n f_i(x)$ — example? ZRM
huge

Stochastic gradient / unbiased estimator randomization of \mathcal{F}_i
 $x_{t+1} = x_t - \eta \tilde{g}_t, \quad \tilde{g}_t = \nabla f_{i_t}(x_t)$

Convergence rates

⊗ non-smooth
 , μ -Lipschitz

convex
 s.l.

Deterministic

$1/\sqrt{\epsilon}$
 $1/\epsilon$

Stochastic

"
 "

⊗ smooth

convex // s.c.

$1/\epsilon // \mu \Leftrightarrow 1/\sqrt{\epsilon} // 1/\epsilon$

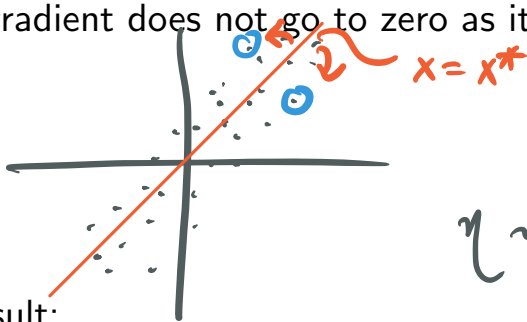
Problem

convergence rate

Why is stochastic gradient not as good as deterministic gradient?

- ▶ In particular not as fast in the smooth case (no self-tuning).
- ▶ Expected gradient does not go to zero as it converges (need small step size).

Illustration:



Convergence result:

$$\mathbb{E} \left[f \left(\frac{1}{T} \sum_{t=1}^T x_t \right) \right] - f(x^*) \leq \frac{R^2}{2\eta T} + \frac{G^2 \eta}{2} \sim o\left(\frac{1}{T}\right) \text{ for } G \rightarrow 0$$

$$\sim o\left(\frac{1}{\sqrt{\epsilon}}\right)$$

$$\eta \sim \sqrt{\epsilon}$$

$$\mathbb{E}[\|g^x\|^2] \leq G^2$$

Trade-off:

- ▶ (stochastic) $O(1/\epsilon)$ iterations but requires 1 gradient per iteration. $o(\sqrt{1/\epsilon})$
- ▶ (deterministic) $O(\log(1/\epsilon))$ iterations but requires n gradients per iteration. $o(d \cdot n \cdot \log(1/\epsilon))$

Mini-batching

Between deterministic and stochastic methods, a common variant is to use mini-batch of sample \mathcal{B}_t to approximate the gradient at x_t

$$\tilde{g}(x_t) = \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \nabla f_i(x_t) \approx \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_t) = \nabla f(x_t)$$

and perform gradient descent with the approximate gradient

$$x_{t+1} = x_t - \eta \tilde{g}(x_t) .$$

I.e., rather than picking a single f_i and using ∇f_i , pick $|\mathcal{B}_t|$ f_i 's and use the average.

Mini-batching

Mini-batch SGD:

$$x_{t+1} = x_t - \eta \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \nabla f_i(x_t) .$$

Unbiased estimator (with uniform sampling):

$$\mathbb{E} \left[\frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \nabla f_i(x_t) \right] = \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \mathbb{E}[\nabla f_i(x_t)] = \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \nabla f(x_t) = \nabla f(x_t)$$

Parallelizable:

- ▶ $|\mathcal{B}|$ gradients can be computed independently in parallel (distributed processing).
- ▶ although requires more work per iteration.

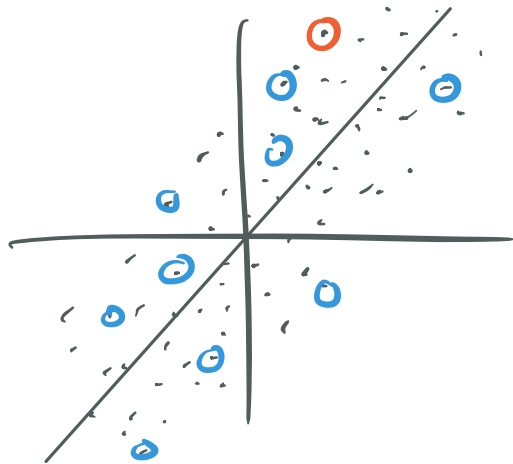
Mini-batching

SGD

mini-batch SGD

$$|B| = 2^k$$

Reduces variance:



- ▶ Although the variance is still not zero (no self-tuning).

Variance reduction

Many forms of variance reduction can exist.

Increasing $|B_t|$

▶ is a form of variance reduction.

▶ A geometric schedule like $|B_{t+1}| = |B_t|/\rho$ can achieve a faster rate.

▶ But this will eventually require $O(n)$ iteration cost.

Other approaches

▶ include control variates, importance sampling, re-parameterization trick, etc.

▶ They improve constants in convergence rate, but not the rate itself unless variance goes to zero.

simplest

stochastic \Rightarrow deterministic

$\rho < 1$ e.g. $\rho = 0.5$

Stochastic average gradient

2012

$x \in \mathbb{R}^d$

Stochastic average gradient or SAG (Schmidt et al. 2017) is one of the very first variance reduction methods that obtain linear rate.

Algorithm:

step (k) :
$$x^{k+1} = x^k - \frac{\eta_k}{n} \sum_{i=1}^n y_i^k, \quad \text{where } y_i^k = \begin{cases} \nabla f_i(x^k) & \text{if } i = i_k \\ y_i^{k-1} & \text{otherwise.} \end{cases}$$

$i = \{1, \dots, i_k, \dots, n\}$
 $\uparrow \uparrow \uparrow \uparrow$

$n \left[\begin{array}{c} \nabla f_1(x^k) \\ \vdots \\ \nabla f_n(x^k) \end{array} \right]_{i_k}$

Idea:

- ▶ Maintain a table of gradients ∇f_i for all i .
- ▶ Update the table with most recent gradient ∇f_{i_k} with randomly selected i_k at each iteration k .
- ▶ Take the parameter update step using the average of these gradients.

i.e., The update step incorporates a gradient with respect to each function like FG, but each iteration only computes the gradient with respect to a single example like SG.

SAG

Pseudo-code (with d to track the quantity $\sum_{i=1}^n y_i$):

Algorithm SAG (step size = α)

Initialize $d = 0$, $y_i = 0$ for $i = 1, 2, \dots, n$

for $k = 0, 1, \dots$ **do**

Sample i from $\{1, 2, \dots, n\}$

Table $\rightarrow d^k = d^{k-1} - y_i + \nabla f_i(x)$

$y_i = \nabla f_i(x)$

parameter $\rightarrow x = x - \frac{\alpha}{n} d$

end for

$$y_i = \nabla f_i(x^{k-1})$$

$$f_i(x) = \frac{1}{2} (a_i^T x - b_i)^2$$

Further notes on SAG:

- ▶ SAG uses a gradient for every example, although the gradients might out of date.
- ▶ SAG requires a memory, but it is possible to reduce memory (e.g. linear models).
- ▶ The stochastic gradient d of SAG is not unbiased ($\mathbb{E}[d^k] \neq \nabla f(x^k)$), but the variance is much reduced from that of the standard stochastic gradient.

Theorem

For ∇f_i is L -continuous, SAG with $\alpha_k = 1/16L$ satisfies

$$\mathbb{E}[f(\bar{x}^k)] - f(x^*) \leq \mathcal{O}\left(\frac{32n}{k}\right).$$

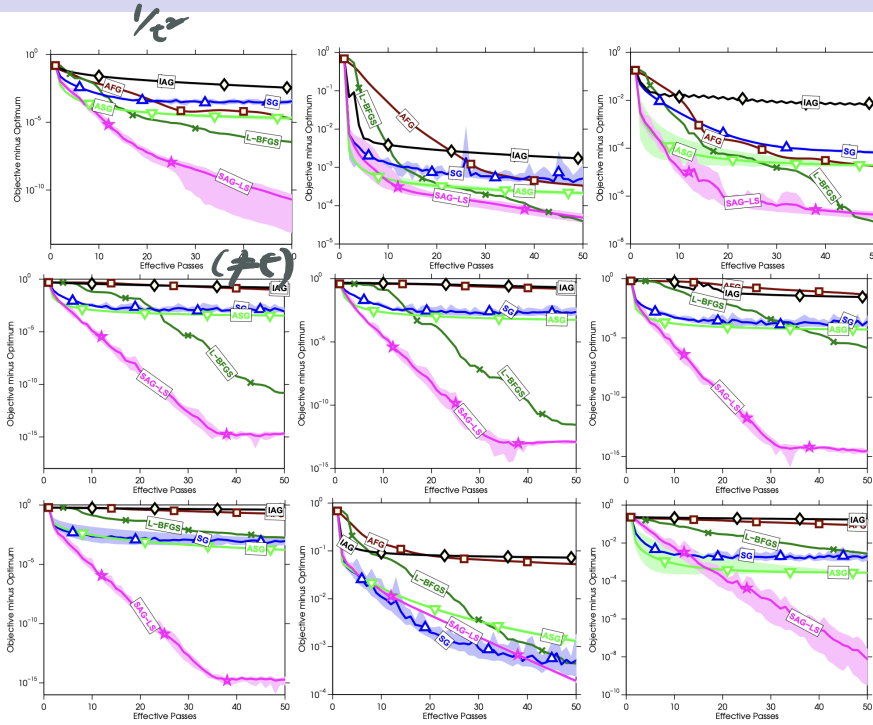
SAG $\mathcal{O}(1/k)$ FG $\mathcal{O}(1/k)$

Further, if f is μ -strongly-convex, then

$\rightarrow (\max\{k, n\}) \log(L/\mu)$

$$\mathbb{E}[f(x^k)] - f(x^*) \leq \mathcal{O}\left(\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8n}\right\}\right)^k\right).$$

- ▶ Despite the low cost of the SAG iterations, with a constant step-size the SAG iterations have an $\mathcal{O}(1/k)$ convergence rate for convex objectives and a linear convergence rate for strongly-convex objectives, like the FG method.
- ▶ SAG achieves convergence rates similar to those of deterministic method; however the constants are different.
- ▶ The proof does not seem to be very straightforward.



Results on binary data sets for L2-regularized logistic regression

Stochastic variance reduced gradient

Another variant is stochastic variance reduced gradient or SVRG (Johnson and Zhang 2013), which gets rid of memory by occasionally computing full gradient.

$$\text{step } t: \quad x_{t+1} = x_t - \eta(\underbrace{\nabla f_{i_t}(x_t) - (\nabla f_{i_t}(y_t) - \nabla f(y_t))}_{\sim \frac{\partial}{\partial t}})$$

where y_t is updated every m iterations.

$(t+m)$

- ▶ SVRG does not store a full table of gradients but just an average and updates it occasionally.
- ▶ It can be shown to reduce variance and achieve convergence rates similar to SAG.
- ▶ The convergence analysis is much simpler.

Stochastic variance reduced gradient

Algorithm SVRG

Initialize \tilde{x}_0

outer for $s = 1, 2, \dots$ do

$\tilde{x} = \tilde{x}_{s-1}$

$\tilde{\mu} = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{x})$ ←

$x_0 = \tilde{x}$

inner for $t = 1, 2, \dots, m$ do

Randomly pick $i_t \in \{1, \dots, n\}$ and update weight

→ $x_t = x_{t-1} - \eta(\nabla f_{i_t}(x_{t-1}) - \nabla f_{i_t}(\tilde{x}) + \tilde{\mu})$ ←

end for

Set $\tilde{x}_s = x_m$

end for

- ▶ (inner loop) same effort as stochastic method
- ▶ (outer loop) full gradient computation

The idea is to reduce variance by recentering: for any two points x and y

$$\tilde{g}(x) = \nabla f_i(x) - (\nabla f_i(y) - \nabla f(y))$$

is a stochastic gradient (*i.e.*, $\mathbb{E}[\tilde{g}(x)] = \nabla f(x)$).

We can develop convergence analysis based on this.

⇒ reduced var.

Lemma

Let f_1, \dots, f_n be β -smooth convex functions, i be a random variable uniformly distributed in $[1, n]$. Then,

$$\mathbb{E}[\|\nabla f_i(x) - \nabla f_i(x^*)\|_2^2] \leq 2\beta(f(x) - f(x^*)) .$$

I.e., the expected value of recentered gradient (squared norm) will decrease as x converges to x^* .

Proof. Let $g_i(x) = f_i(x) - f_i(x^*) - \nabla f_i(x^*)^\top (x - x^*)$. By convexity of f_i one has $g_i(x) \geq 0$. Using the progress bound for smooth functions yields $\|\nabla g_i(x)\|_2^2 \leq 2\beta g_i(x)$, which can be written as

$$\|\nabla f_i(x) - \nabla f_i(x^*)\|_2^2 \leq 2\beta(f_i(x) - f_i(x^*) + \nabla f_i(x^*)^\top (x - x^*)) .$$

Taking expectation will finish the proof.

Theorem

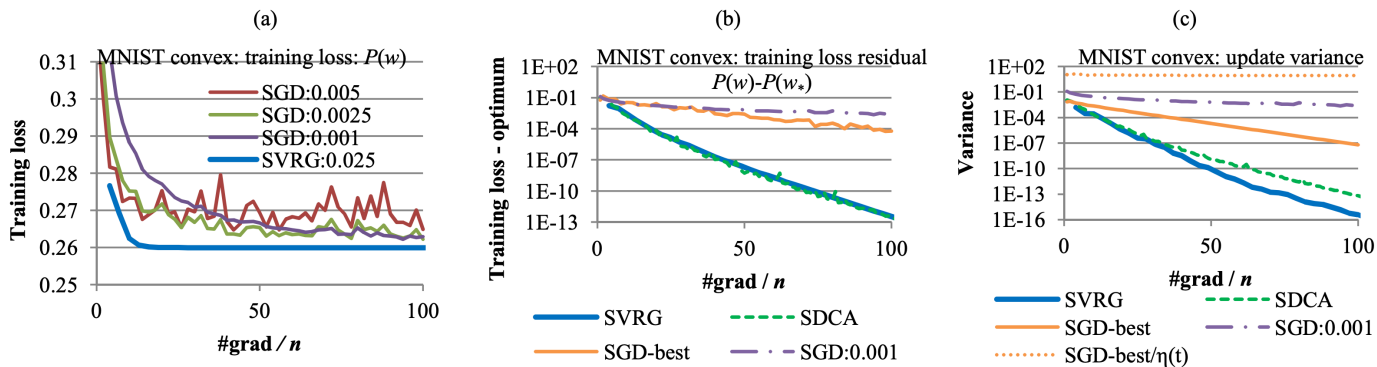
Let f_1, \dots, f_n be β -smooth convex functions and f be α -strongly convex. Then, SVRG with $\eta = 1/10\beta$ and $k = 20(\beta/\alpha)$ satisfies

$$\mathbb{E}f(y^{s+1}) - f(x^*) \leq 0.9^s (f(y^{(1)}) - f(x^*)) .$$

Proof is referred to Bubeck et al. [2015](#).

Note:

- ▶ This result shows linear convergence rate.
- ▶ SVRG requires $n + m$ gradient computations where m depends on κ .



Multiclass logistic regression (convex) on MNIST.




Thank you

Any questions?

A lot of material in this course is borrowed or derived from the following:

- ▶ Numerical Optimization, Jorge Nocedal and Stephen J. Wright.
- ▶ Convex Optimization, Stephen Boyd and Lieven Vandenberghe.
- ▶ Convex Optimization, Ryan Tibshirani.
- ▶ Optimization for Machine Learning, Martin Jaggi and Nicolas Flammarion.
- ▶ Optimization Algorithms, Constantine Caramanis.
- ▶ Advanced Machine Learning, Mark Schmidt.

References I

-  Bubeck, Sébastien et al. (2015). “Convex optimization: Algorithms and complexity”. In: *Foundations and Trends® in Machine Learning* 8.3-4, pp. 231–357.
-  Johnson, Rie and Tong Zhang (2013). “Accelerating stochastic gradient descent using predictive variance reduction”. In: *Advances in neural information processing systems* 26.
-  Schmidt, Mark, Nicolas Le Roux, and Francis Bach (2017). “Minimizing finite sums with the stochastic average gradient”. In: *Mathematical Programming* 162.1, pp. 83–112.