ECE 8101: Nonconvex Optimization for Machine Learning

Lecture Note 2-5: Variance-Reduced First-Order Methods

Jia (Kevin) Liu

Assistant Professor
Department of Electrical and Computer Engineering
The Ohio State University, Columbus, OH, USA

Spring 2022

Outline

In this lecture:

- Key Idea of Variance-Reduced Methods
- SAG, SVRG, SAGA, SPIDER/SpiderBoost, SARAH, and PAGE
- Convergence results

Recap: Stochastic Gradient Descent

- SGD Convergence Performace
  - Constant step-size: SGD converges quickly to an approximation
    - ★ Step-size s and batch size B, converges to a $\frac{s\sigma^2}{B}$-error ball
  - Decreasing step-size: SGD converges slowly to exact solution
- Two "control knobs" to improve SGD convergence performance
  - Decrease (gradually) step-sizes:
    - ★ Improves convergence accuracy
    - ★ Make convergence too slow
  - Increase batch-sizes:
    - ★ Leads to faster rate of iterations
    - ★ Makes setting step-sizes easier
    - ★ But increases the iteration cost
- Question: Could we achieve fast convergence rate with small batch-size?

Stochastic Average Gradient (SAG)

- Growing batch-size B_k eventually requires $O(N)$ samples per iteration
- Question: Can we achieve one sample per iteration and same iteration complexity as deterministic first-order methods?
- Answer: Yes, the first method was the stochastic average gradient (SAG) method [Le Roux et al. 2012]
- To understand SAG, it's insightful to view GD as performing the following iteration in solving the finite-sum problem:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{s_k}{N} \sum_{i=1}^{N} \mathbf{v}_k^i$$

  where in each step we set $\mathbf{v}_k^i = \nabla f_i(\mathbf{x}_k)$ for all i
- SAG method: Only set $\mathbf{v}_k^{i_k} = \nabla f_{i_k}(\mathbf{x}_k)$ for randomly chosen i_k
  - All other $\mathbf{v}_k^{i_k}$ are kept at their previous values (a lazy update approach)

Stochastic Average Gradient (SAG)

- One can think of SAG as having a memory:

$$\nabla f(x_k) = \begin{bmatrix} \text{———} & \mathbf{v}^1 & \text{———} \\ \text{———} & \mathbf{v}^2 & \text{———} \\ & \vdots & \\ \text{———} & \mathbf{v}^N & \text{———} \end{bmatrix},$$

  where $\mathbf{v}^i$ is the gradient $\nabla f_i(\mathbf{x}_{k'})$ from the last k' where i is selected

- In each iteration:
  - Randomly choose one of the $\mathbf{v}^i$ and update it to the current gradient
  - Take a step in the direction of the average of these $\mathbf{v}^i$

Stochastic Average Gradient (SAG)

- Basic SAG algorithm (maintains $\mathbf{g} = \sum_{i=1}^{N} \mathbf{v}^i$):
  - Set $\mathbf{g} = \mathbf{0}$ and gradient approximation $\mathbf{v}^i = \mathbf{0}$ for $i = 1, \ldots, N$.
  - while (1):
    1. Sample i from $\{1, 2, \ldots, N\}$
    2. Compute $\nabla f_i(\mathbf{x})$
    3. $\mathbf{g} = \mathbf{g} - \mathbf{v}^i + \nabla f_i(\mathbf{x})$
    4. $\mathbf{v}^i = \nabla f_i(\mathbf{x})$
    5. $\mathbf{x}^+ = \mathbf{x} - \frac{s}{N}\mathbf{g}$

- Iteration cost is $O(d)$ (one sample)

- Memory complexity is $O(Nd)$
  - Could be less if the model is sparse
  - Could reduce to $O(N)$ for linear models $f_i(\mathbf{x}) = h(\mathbf{x}^\top \boldsymbol{\xi}^i)$:

$$\nabla f_i(\mathbf{x}) = \underbrace{h'(\mathbf{x}^\top \boldsymbol{\xi}^i)}_{\text{scalar}} \underbrace{\mathbf{x}^i}_{\text{data}}$$

  - But for neural networks, would still need to store all activations (typically impractical)

Stochastic Average Gradient (SAG)

- The SAG algorithm:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{s_k}{N} \sum_{i=1}^{N} \mathbf{v}_k^i,$$

where in each iteration, $\mathbf{v}_k^{i_k} = \nabla f_{i_k}(\mathbf{x}_k)$ for a randomly chosen i_k

- Unlike batching in SGD, use a "gradient" for every sample
  - But the gradient might be out of date due to lazy update
- Intuition: $\mathbf{v}_k^i \to \nabla f_i(\mathbf{x}^*)$ at the same rate that $\mathbf{x}_k \to \mathbf{x}^*$
  - so the variance $\|\mathbf{e}_k\|^2$ ("bad term") converges linearly to 0

Convergence Rate of SAG

Theorem 1 ([Le Roux et al. 2012])

If each ∇f_i is L-Lipschitz continuous and f is strongly convex, with $s_k = 1/16L$, SAG satisfies:

$$\mathbb{E}[f(\mathbf{x}_k) - f^*] = O\left(\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^{k}\right)$$

- Sample Complexity: Number of ∇f_i evaluations to reach accuracy ϵ:
  - Stochastic: $O(\frac{L}{\mu}(1/\epsilon))$
  - Gradient: $O(n\frac{L}{\mu}\log(1/\epsilon))$
  - Nesterov: $O(n\sqrt{\frac{L}{\mu}}\log(1/\epsilon))$
  - SAG: $O(\max\{n, \frac{L}{\mu}\}\log(1/\epsilon))$

- Note: L values are different between algorithms

Stochastic Variance-Reduced Gradient (SVRG)

Idea: Get rid of memory by periodically computing full gradient [Johnson&Zhang,'13]

- Start with some $\tilde{\mathbf{x}}^0 = \mathbf{x}_m^0 = \mathbf{x}_0$, where m is a parameter. Let $S = \lceil T/m \rceil$
- for $s = 0, 1, 2, \ldots, S-1$
  - $\mathbf{x}_0^{s+1} = \mathbf{x}_m^s$
  - $\nabla f(\tilde{\mathbf{x}}^s) = \frac{1}{N}\sum_{i=1}^N \nabla f_i(\tilde{\mathbf{x}}^s)$
  - for $k = 0, 1, 2, \ldots, m-1$
    - ★ Uniformly pick a batch $I_k \subset \{1, 2, \ldots, N\}$ at random (with replacement), with batch size $|I_k| = B$
    - ★ Let $\mathbf{v}_k^{s+1} = \frac{1}{B}\sum_{i=1}^B [\nabla f_{i_k}(\mathbf{x}_k^{s+1}) - \nabla f_{i_k}(\tilde{\mathbf{x}}^s)] + \nabla f(\tilde{\mathbf{x}}^s)$
    - ★ $\mathbf{x}_{k+1}^{s+1} = \mathbf{x}_k^{s+1} - s_k \mathbf{v}_k^{s+1}$
  - $\tilde{\mathbf{x}}^{s+1} = \mathbf{x}_m^{s+1}$
- Output: Chose $\mathbf{x}_a$ uniformly at random from $\{\{\mathbf{x}_k^{s+1}\}_{k=0}^{m-1}\}_{s=0}^{S-1}$

Convex settings: Convergence properties similar to SAG for suitable q

- Unbiased: $\mathbb{E}[\cancel{\nabla f_{i_k}(\mathbf{x}_s)}\ \mathbf{v}_k^{s+1}] = \nabla f(\mathbf{x}_k^{s+1})$ $\qquad \mathbb{E}[\mathbf{v}_k^{s+1}] = \nabla f(\mathbf{x}_k^{s+1})$
- Theoretically $\cancel{q}$ m depends on L, μ, and N ($\cancel{q}$ $m = N$ works well empirically)
- $O(d)$ storage complexity (2B+1 gradients per iteration on average)
- Last step $\tilde{\mathbf{x}}^{s+1}$ in outer loop can be randomly chosen from inner loop iterates

Convergence Rate of SVRG (Nonconvex)

- Consider finite-sum problem $\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \triangleq \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x})$, where both $f(\cdot)$ and $f_i(\cdot)$ are nonconvex, differentiable, and L-smooth.
- Define a sequence $\{\Gamma_k\}$ with $\Gamma_k \triangleq s_k - \frac{c_{k+1}s_k}{\beta_k} - s_k^2 L - 2c_{k+1}s_k^2$, where parameters c_{k+1} and β_k are TBD shortly.

Theorem 2 ([Reddi et al. '16])

Let $c_m = 0$, $s_k = s > 0$, $\beta_k = \beta > 0$, *and* $c_k = c_{k+1}(1 + s\beta + 2s^2L^2/B) + s^2L^3/B$ *such that* $\Gamma_k > 0$ *for* $k = 0, \ldots, m-1$. *Let* $\gamma = \min_k \Gamma_k$. *Also, let* T *be a multiple of* m. *Then, the output* $\mathbf{x}_a$ *of SVRG satisfies:*

$$\mathbb{E}[\|\nabla f(\mathbf{x}_a)\|^2] \le \frac{f(\mathbf{x}_0) - f^*}{T\gamma}. = O(\tfrac{1}{T}).$$

Theorem 2 ([Reddi et al. '16])

Let $c_m = 0$, $s_k = s > 0$, $\beta_k = \beta > 0$, and $c_k = c_{k+1}(1 + s\beta + 2s^2L^2/B) + s^2L^3/B$ such that $\Gamma_k > 0$ for $k = 0, \dots, m-1$. Let $\gamma = \min_k \Gamma_k$. Also, let T be a multiple of m. Then, the output $\mathbf{x}_a$ of SVRG satisfies:

$$\mathbb{E}[\|\nabla f(\mathbf{x}_a)\|^2] \le \frac{f(\mathbf{x}_0) - f^*}{T\gamma}. = O\left(\frac{1}{T}\right).$$

Proof: ① Lemma 1: Define a "Lyapunov" fn: $R_k^{s+1} \triangleq \mathbb{E}\left[f(\underline{x}_k^{s+1}) + c_k \|\underline{x}_k^{s+1} - \tilde{\underline{x}}^s\|^2\right]$

For $c_k, c_{k+1}, \beta_k > 0$, suppose we have the following:

$$c_k = c_{k+1}\left(1 + s_k\beta_k + \frac{2s_k^2L^2}{B}\right) + \frac{s_k^3L^3}{B}, \quad k = 0, \dots, m-1.$$

Let s_k, β_k, c_k be chosen s.t. $\Gamma_k > 0$, Then $\{\underline{x}_k^{s+1}\}$ satisfies:

$$\mathbb{E}\left[\|\nabla f(\underline{x}_k^{s+1})\|^2\right] \le \frac{R_k^{s+1} - R_{k+1}^{s+1}}{\Gamma_k}.$$

lemma:

①-1°

Proof of Lemma 1: Since f is L-smooth, we have from descent

$$\mathbb{E}\left[f(\underline{x}_{k+1}^{s+1})\right] \le \mathbb{E}\left[f(\underline{x}_k^{s+1}) + \nabla f(\underline{x}_k^{s+1})^T \underbrace{(\underline{x}_{k+1}^{s+1} - \underline{x}_k^{s+1})}_{= s_k \underline{v}_k^{s+1}} + \frac{L}{2}\|\underline{x}_{k+1}^{s+1} - \underline{x}_k^{s+1}\|^2\right] \quad (1)$$

Using SVRG update and also the unbiasedness: $\mathbb{E}[\underline{v}_k^{s+1}] = \nabla f(\underline{x}_k^{s+1})$.

$$\mathbb{E}\left[f(\underline{x}_{k+1}^{s+1})\right] \le \mathbb{E}\left[f(\underline{x}_k^{s+1}) - s_k\|\nabla f(\underline{x}_k^{s+1})\|^2 + \frac{Ls_k^2}{2}\underbrace{\|\underline{v}_k^{s+1}\|^2}_{\text{drift. vector}}\right] \quad (2)$$

(similar to derivations of Thm 2 is SGD)

Consider the Lyapunov fn: $R_k^{s+1} = \mathbb{E}\left[f(\underline{x}_k^{s+1}) + c_k\|\underline{x}_k^{s+1} - \tilde{\underline{x}}^s\|^2\right]$

Next, we will analyze 1-step Lyapunov drift: $R_{k+1}^{s+1} - R_k^{s+1}$.

To do so, we first bnd $\mathbb{E}\left[\|x_{k+1}^{s+1}-\tilde{x}\|^2\right]$:

$$\mathbb{E}\left[\|x_{k+1}^{s+1}-\tilde{x}^s\|^2\right] \overset{\text{add \& subtract } x_k^{s+1}}{=\!=} \mathbb{E}\left[\|x_{k+1}^{s+1}-x_k^{s+1}+x_k^{s+1}-\tilde{x}^s\|^2\right]$$

$$= \mathbb{E}\left[\|x_{k+1}^{s+1}-x_k^{s+1}\|^2 + \|x_k^{s+1}-\tilde{x}^s\|^2 + 2\underbrace{\langle x_{k+1}^{s+1}-x_k^{s+1}, x_k^{s+1}-\tilde{x}^s\rangle}_{\downarrow \text{ unbiasedness of SVRG}}\right]$$

$$\overset{\downarrow \text{SVRG}}{=} \mathbb{E}\left[s_k^2\|v_k^{s+1}\|^2 + \|x_k^{s+1}-\tilde{x}^s\|^2\right] - 2s_k\mathbb{E}\left[\underbrace{\langle \nabla f(x_k^{s+1}), x_k^{s+1}-\tilde{x}^s\rangle}_{\text{Fenchel-Young's Ineq.}}\right]$$

$$\le \mathbb{E}\left[s_k^2\|v_k^{s+1}\|^2 + \|x_k^{s+1}-\tilde{x}^s\|^2\right] - 2s_k\mathbb{E}\left[\frac{1}{2\beta_k}\|\nabla f(x_k^{s+1})\|^2 + \frac{\beta_k}{2}\|x_k^{s+1}-\tilde{x}^s\|^2\right] \qquad (3).$$

Plugging (2) and (3) into R_{k+1}^{s+1} to obtain:

$$R_{k+1}^{s+1} = \mathbb{E}\left[\underbrace{f(x_{k+1}^{s+1})}_{(2)} + c_{k+1}\underbrace{\|x_{k+1}^{s+1}-\tilde{x}^s\|^2}_{(3)}\right]$$

$$\le \mathbb{E}\left[f(x_k^{s+1}) - s_k\|\nabla f(x_k^{s+1})\|^2 + \frac{Ls_k^2}{2}\|v_k^{s+1}\|^2\right]$$

$$+ \mathbb{E}\left[c_{k+1}s_k^2\|v_k^{s+1}\|^2 + c_{k+1}\|x_k^{s+1}-\tilde{x}^s\|^2 + \right.$$

$$\left. 2c_{k+1}s_k\mathbb{E}\left[\frac{1}{2\beta_k}\|\nabla f(x_k^{s+1})\|^2 + \frac{\beta_k}{2}\|x_k^{s+1}-\tilde{x}^s\|^2\right]\right]$$

$$= \mathbb{E}\left[f(x_k^{s+1})\right] - \left(s_k - \frac{c_{k+1}s_k}{\beta_k}\right)\mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right] + \left(\frac{Ls_k^2}{2} + c_{k+1}s_k^2\right)\mathbb{E}\left[\|v_k^{s+1}\|^2\right]$$

$$+ \left(c_{k+1} + c_{k+1}s_k\beta_k\right)\mathbb{E}\left[\|x_k^{s+1}-\tilde{x}^s\|^2\right] \qquad (4).$$

①-2°: Claim: $\mathbb{E}\left[\|v_k^{s+1}\|^2\right] \le 2\,\mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right] + \frac{2L^2}{B}\,\mathbb{E}\left[\|x_k^{s+1} - \tilde{x}^s\|^2\right]$

Proof: Let $\delta_k^{s+1} = \frac{1}{B}\sum_{i \in I_k}\left(\nabla f_{i_k}(x_k^{s+1}) - \nabla f_{i_k}(\tilde{x}^s)\right)$.

Note: $\nabla f(x_k^{s+1}) = \mathbb{E}\left[\delta_k^{s+1} + \nabla f(\tilde{x}^s)\right]$ (unbiasedness).

From definition of v_k^{s+1}:

$$\mathbb{E}\left[\|v_k^{s+1}\|^2\right] = \mathbb{E}\left[\|\delta_k^{s+1} + \nabla f(\tilde{x}^s)\|^2\right]$$

add & subtract $\nabla f(x_k^{s+1})$

$$\mathbb{E}\left[\|\delta_k^{s+1} + \underbrace{\nabla f(\tilde{x}^s) - \nabla f(x_k^{s+1})}_{-\mathbb{E}[\delta_k^{s+1}]} + \nabla f(x_k^{s+1})\|^2\right]$$

$$\le 2\,\mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right] + 2\,\mathbb{E}\left[\|\delta_k^{s+1} - \mathbb{E}[\delta_k^{s+1}]\|^2\right]$$

$\mathbb{E}\left[\|z_1 + \cdots + z_r\|^2\right] \le r\,\mathbb{E}\left[\|z_1\|^2 + \cdots + \|z_r\|^2\right]$

$$= 2\,\mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right] + \frac{2}{B^2}\,\mathbb{E}\left[\Big\|\sum_{i_k \in I_k}\big(\underbrace{\nabla f_{i_k}(x_k^{s+1}) - \nabla f_{i_k}(\tilde{x}^s)}_{\delta_k^{s+1}} - \mathbb{E}[\delta_k^{s+1}]\big)\Big\|^2\right]$$

$$\le 2\,\mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right] + \frac{2}{B^2}\,\mathbb{E}\left[\sum_{i_k \in I_k}\|\underbrace{\nabla f_{i_k}(x_k^{s+1}) - \nabla f_{i_k}(\tilde{x}^s)}_{\delta_k^{s+1}} - \mathbb{E}[\delta_k^{s+1}]\|^2\right]$$

$\mathbb{E}\left[\|\xi - \mathbb{E}\xi\|^2\right] \le \mathbb{E}\left[\|\xi\|^2\right]$

$\mathbb{E}\left[\|z_1 + \cdots + z_r\|^2\right] \le \mathbb{E}\left[\|z_1\|^2 + \cdots + \|z_r\|^2\right]$ if $z_1 \cdots z_r$ are indep. with mean 0

$$\le 2\,\mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right] + \frac{2}{B^2}\,\mathbb{E}\left[\sum_{i_k \in I_k}\|\underbrace{\nabla f_{i_k}(x_k^{s+1}) - \nabla f_{i_k}(\tilde{x}^s)}_{\le L\|x_k^{s+1} - \tilde{x}^s\|}\|^2\right]$$

$$\le 2\,\mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right] + \frac{2}{B^2}\cdot B\cdot L^2\,\mathbb{E}\left[\|x_k^{s+1} - \tilde{x}^s\|^2\right].$$ The claim is proved.

Using the Claim in (4)

$$\underbrace{R_{k+1}^{s+1} \le \mathbb{E}\left[f(x_k^{s+1})\right] - \Big(\underbrace{s_k - \frac{c_{k+1} s_k}{\beta_k} - s_k^2 L - 2c_{k+1} s_k^2}_{\triangleq \Gamma_k}\Big)\mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right]}$$

$$+ \Big[\underbrace{c_{k+1}\big(1 + s_k\beta_k + 2\frac{s_k^2 L^2}{\beta}\big) + \frac{s_k^2 L^3}{\beta}}_{\triangleq c_k}\Big]\mathbb{E}\left[\|x_k^{s+1} - \tilde{x}^s\|^2\right] = R_k^{s+1}$$

$$\Rightarrow \mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right] \le \frac{R_k^{s+1} - R_{k+1}^{s+1}}{\Gamma_k}.$$ Lemma 1 is proved. ∎

To complete the proof of Thm 2:

Since $s_k = s,\ \forall k$, using Lemma 1 and telescoping sum (inner loop).

$$\sum_{k=0}^{m-1} \mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right] \le \frac{R_0^{s+1} - R_m^{s+1}}{\gamma}.$$

$$R_m^{s+1} \overset{\text{def of } c_m = 0}{=} \mathbb{E}\left[f(x_m^{s+1})\right] = \mathbb{E}\left[f(\tilde{x}^{s+1})\right]$$

$$R_0^{s+1} = \mathbb{E}\left[f(\tilde{x}^s)\right] \quad (\text{since } x_0^{s+1} = \tilde{x}^s).$$

Summing over all epochs yields:

$$\frac{1}{T}\sum_{s=0}^{S-1}\sum_{k=0}^{m-1} \mathbb{E}\left[\|\nabla f(x_k^{s+1})\|^2\right] \le \frac{f(x_0) - f^*}{T\gamma}.$$ ∎

Let $s = \frac{\mu_0}{LN^\alpha}$, where $\mu_0 \in (0,1)$ and $\alpha \in (0,1]$, $\beta = L/N^{\frac{\alpha}{2}}$

$m = \lfloor N^{\frac{3\alpha}{2}}/(3\mu_0) \rfloor$, T is some multiple of m. Then, $\exists$ constants $\mu_0, \nu > 0$, s.t. we have $\gamma \ge \frac{\nu}{LN^\alpha}$, and

$$\mathbb{E}\left[\|\nabla f(\underline{x}_a)\|^2\right] \leq \frac{LN^{\alpha}(f(x_0)-f^*)}{T\nu} \leq \varepsilon^2$$

$\Rightarrow$ Sample complexity $\begin{cases} O\left(N + \left(N^{1-\frac{\alpha}{2}}/\varepsilon^2\right)\right), & \text{if } \alpha \leq \frac{2}{3} \\ O\left(N + N^{\alpha}/\varepsilon^2\right) & \text{if } \alpha > \frac{2}{3} \end{cases}$

If $\alpha = \frac{2}{3}$, $\Rightarrow$ $O\left(N + N^{\frac{2}{3}} \underset{\substack{\uparrow \\ f(x_0)-f^*}}{\Delta_0} \varepsilon^{-2}\right)$. (GD: $N\varepsilon^{-2}$)

SAGA (SAG Again?)

Basic SAGA algorithm [Defazio et al. 2014]: Similar in spirit to SAG

- Initialize $\mathbf{x}_0$; Create a table, containing gradients and $\mathbf{v}_0^i = \nabla f_i(\mathbf{x}_0)$
- In iterations $k = 0, 1, 2, \ldots$:
  1. Pick a random $i_k \in \{1, \ldots, N\}$ uniformly at random and compute $\nabla f_{i_k}(\mathbf{x}_k)$.
  2. Update $\mathbf{x}_{k+1}$ as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \left(\nabla f_{i_k}(\mathbf{x}_k) - \mathbf{v}_k^{i_k} + \frac{1}{N} \sum_{i=1}^{N} \mathbf{v}_k^i \right)$$

  3. Update table entry $\mathbf{v}_{k+1}^{i_k} = \nabla f_i(\mathbf{x}_k)$. Set all other $\mathbf{v}_{k+1}^i = \mathbf{v}_k^i$, $i \neq i_k$, i.e., other table entries remain the same

SAGA (SAG Again?)

$N^{\frac{2}{3}}\epsilon^{-2}$

- SAGA basically matches convergence rates of SAG (for both convex and strongly convex cases), but the proof is simpler (due to unbiasedness)

- Another strength of SAGA is that it can extend to composite problems:

$$\min_{\mathbf{x}} \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x}) + h(\mathbf{x}),$$

where each $f_i(\cdot)$ is L-smooth, and h is convex and non-smooth, but has a known proximal operator

$$\mathbf{x}_{k+1} = \mathrm{prox}_{h,s_k}\left\{\mathbf{x}_k - s_k\left(\nabla f_{i_k}(\mathbf{x}_k) - \mathbf{v}_k^{i_k} + \frac{1}{N}\sum_{i=1}^{N}\mathbf{v}_k^i\right)\right\}.$$

But it is unknown whether SAG is convergent or not under proximal operator

SAGA Variance Reduction

- Stochastic gradient in SAGA:

$$\underbrace{\nabla f_{i_k}(\mathbf{x}_k)}_{X} - \underbrace{\left(\mathbf{v}_k^{i_k} - \frac{1}{N}\sum_{i=1}^{N}\mathbf{v}_k^{i}\right)}_{Y}$$

- Note: $\mathbb{E}[X] = \nabla f(\mathbf{x}_k)$ and $\mathbb{E}[Y] = 0 \Rightarrow$ we have an unbiased estimator
- Note: $X - Y \to 0$ as $k \to \infty$, since $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$ converges to some $\bar{\mathbf{x}}$, the difference between the first two terms converges to zero. The last term converges to gradient at stationarity, i.e., also zero
- Thus, the overall ℓ_2 norm estimator (i.e., variance) decays to zero

Comparisons between SAG, SVRG, and SAGA

A general variance reduction approach: Want to estimate $\mathbb{E}[X]$

- Suppose we can compute $\mathbb{E}[Y]$ for a r.v. Y that is highly correlated with X
- Consider the estimator θ_α as an approximation to $\mathbb{E}[X]$:

$$\theta_\alpha \triangleq \alpha(X - Y) + \mathbb{E}[Y], \text{ for some } \alpha \in (0, 1]$$

- Observations:
  - $\mathbb{E}[\theta_\alpha] = \alpha\mathbb{E}[X] + (1-\alpha)\mathbb{E}[Y]$, i.e., a convex combination of $\mathbb{E}[X]$ and $\mathbb{E}[Y]$.
  - Standard VR: $\alpha = 1$ and hence $\mathbb{E}[\theta_\alpha] = \mathbb{E}[X]$
  - Variance of θ_α: $\mathrm{Var}(\theta_\alpha) = \alpha^2[\mathrm{Var}(X) + \mathrm{Var}(Y) - 2\mathrm{Cov}(X, Y)]$
  - If $\mathrm{Cov}(X, Y)$ is large, variance of θ_α is reduced compared to X
  - Letting α from 0 to 1, $\mathrm{Var}(X) \uparrow$ to max value while decreasing bias to zero

    bias-variance trade-off.

- SAG, SVRG, and SAGA can be derived from this VR viewpoint:
  - SAG: Let $X = \nabla f_{i_k}(\mathbf{x}_k)$ and $Y = \mathbf{v}_k^{i_k}$, $\alpha = 1/N$ (biased)
  - SAGA: Let $X = \nabla f_{i_k}(\mathbf{x}_k)$ and $Y = \mathbf{v}_k^{i_k}$, $\alpha = 1$ (unbiased)
  - SVRG: Let $X = \nabla f_{i_k}(\mathbf{x}_k)$ and $Y = \nabla f_{i_k}(\tilde{\mathbf{x}})$ (unbiased), $\alpha = 1$.
  - Variance of SAG is $1/N^2$ times of that of SAGA

Comparisons between SAG, SVRG, and SAGA

- Update rules:

$$\text{(SAG)} \qquad \mathbf{x}_{k+1} = \mathbf{x}_k - s\left[\frac{1}{N}(\nabla f_{i_k}(\mathbf{x}_k) - \mathbf{v}_k^{i_k}) + \frac{1}{N}\sum_{i=1}^{N}\mathbf{v}_k^i\right]$$

$$\text{(SAGA)} \qquad \mathbf{x}_{k+1} = \mathbf{x}_k - s\left[\nabla f_{i_k}(\mathbf{x}_k) - \mathbf{v}_k^{i_k} + \frac{1}{N}\sum_{i=1}^{N}\mathbf{v}_k^i\right]$$

$$\text{(SVRG)} \qquad \mathbf{x}_{k+1} = \mathbf{x}_k - s\left[\nabla f_{i_k}(\mathbf{x}_k) - \nabla f_{i_k}(\tilde{\mathbf{x}}) + \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\tilde{\mathbf{x}})\right]$$

- SVRG: $\tilde{\mathbf{x}}$ is not updated very step (only updated in the start of outer loops)
- SAG & SAGA: Update $\mathbf{v}_k^{i_k}$ each time index i_k is picked
- SVRG vs. SAGA:
  - SVRG: Low memory cost, slower convergence (same convergence rate order)
  - SAGA: High memory cost, faster convergence
- SAGA can be viewed as a midpoint between SAG and SVRG

Stochastic Recursive Gradient Algorithm (SARAH)

GD: $\frac{C}{T} \le \epsilon^2 \Rightarrow O(N\epsilon^{-2})$ SGD: $\frac{C}{\sqrt{T}} \le \epsilon^2 \Rightarrow O(\epsilon^{-4})$.

- Sample complexity of GD, SGD, SVRG, and SAGA for ϵ-stationarity:
  - GD and SGD require $O(N\epsilon^{-2})$ and $O(\epsilon^{-4})$, respectively[1]
  - $B = 1$: Both SVRG and SARAH guarantee only $O(N\epsilon^{-2})$, same as GD
  - $B = N^{\frac{2}{3}}$: Both SVRG and SAGA achieve $O(N^{\frac{2}{3}}\epsilon^{-2})$, $N^{\frac{1}{3}}$ times better than GD in terms of dependence on N

- However, the sample complexity lower bound is $\Omega(\sqrt{N}\epsilon^{-2})$
  - There exist sample complexity order-optimal algorithms (e.g., SPIDER [Fang et al. 2018] and PAGE [Li et al. 2020])

- These order-optimal algorithms are variants of SARAH [Nguyen et al. 2017]
  - Sample complexity for convex and strongly convex problems: $O(N + 1/\epsilon^2)$ and $O((N+\kappa)\log(1/\epsilon))$, respectively ($\kappa = L/\mu$, a single outer loop)
  - Sample complexity for nonconvex problems: $O(N + L^2/\epsilon^{+4})$ (step size $s = O(1/L\sqrt{T})$, non-batching, a single outer loop)

[1] For simplicity, we ignore all other parameters except N and ϵ here.

Stochastic Recursive Gradient Algorithm (SARAH)

The SARAH algorithm:

- Pick learning rate $s > 0$ and inner loop size m
- for $s = 0, 1, 2, \ldots, S-1$
  - $\mathbf{x}_0^{s+1} = \tilde{\mathbf{x}}^s$
  - $\mathbf{v}_0^{s+1} = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\mathbf{x}_0^{s+1})$
  - $\mathbf{x}_1^{s+1} = \mathbf{x}_0^{s+1} - s\mathbf{v}_0^{s+1}$
  - for $k = 1, 2, \ldots, m-1$
    - Uniformly pick a batch $I_k \subset \{1, 2, \ldots, N\}$ at random (with replacement), with batch size $|I_k| = B$
    - Let $\mathbf{v}_k^{s+1} = \frac{1}{B}\sum_{i \in I_k} [\nabla f_{i_k}(\mathbf{x}_k^{s+1}) - \nabla f_{i_k}(\mathbf{x}_{k-1}^{s+1})] + \mathbf{v}_{k-1}^{s+1}$
    - $\mathbf{x}_{k+1}^{s+1} = \mathbf{x}_k^{s+1} - s\mathbf{v}_k^{s+1}$
  - $\tilde{\mathbf{x}}^{s+1} = \mathbf{x}_k^{s+1}$ with k chosen uniformly at random from $\{0, 1, \ldots, m\}$
- Output: Chose $\mathbf{x}_a$ uniformly at random from $\{\{\mathbf{x}_k^{s+1}\}_{k=0}^{m-1}\}_{s=0}^{S-1}$

Comparison to SVRG (ignoring outer loop index s):

- SVRG: $\mathbf{v}_k = \nabla f_{i_k}(\mathbf{x}_k) - \nabla f_{i_k}(\mathbf{x}_0) + \mathbf{v}_0$ (unbiased)
- SARAH: $\mathbf{v}_k = \nabla f_{i_k}(\mathbf{x}_k) - \nabla f_{i_k}(\mathbf{x}_{k-1}) + \mathbf{v}_{k-1}$ (biased)

SPIDER/SpiderBoost

- SPIDER [Fang et al. 2018]: Provides the first sample complexity lower bound and the first sample complexity order-optimal algorithm
  - SPIDER stands for "stochastic path-integrated differential estimator"
  - Lower bound is for small data regime $N = O(L^2(f(\mathbf{x}_0) - f^*)\epsilon^{-4})$
  - Sample complexity: $\Omega(\sqrt{N}\epsilon^{-2})$
  - However, requires very small step-size $O(\epsilon/L)$, poor convergence in practice
  - Original proof of SPIDER is technically too complex and hence hard to generalize the method to composite optimization problems

- SpiderBoost [Wang et al. 2018] [Wang et al. NeurIPS'19]:
  - Same algorithm, same sample complexity, but relax the step-size to $O(1/L)$
  - Simpler proof and can be generalized to composite optimization problems
  - Also works well with heavy-ball momentum

SPIDER/SpiderBoost

The SPIDER/SpiderBoost Algorithm

- Pick learning rate $s = 1/2L$, epoch length ~~T~~ m, starting point $\mathbf{x}_0$, batch size B, number of iteration T
- **for** $k = 0, 1, 2, \dots, T-1$

    if $k \mod m = 0$ **then**

        Compute full gradient $\mathbf{v}_k = \nabla f(\mathbf{x}_k)$

    else

        Uniformly randomly pick $I_k \subset \{1, \dots, N\}$ (with replacement) with $|I_k| = B$. Compute

$$\mathbf{v}_k = \frac{1}{B} \sum_{i \in I_k} [\nabla f_i(\mathbf{x}_k) - \nabla f_i(\mathbf{x}_{k-1})] + \mathbf{v}_{k-1}$$

    end if

    Let $\mathbf{x}_{k+1} = \mathbf{x}_k - s\mathbf{v}_k$

  end for

  Output: $\mathbf{x}_\xi$, where ξ is picked uniformly at random from $\{0, \dots, T-1\}$

Probabilistic Gradient Estimator (PAGE)

- SPIDER/SpiderBoost: Sample complexity LB is for small data regime
- PAGE [Li et al. ICML'21]: Proved the same lower bound $\Omega(\sqrt{N}\epsilon^{-2})$ without any assumption on data set size N and provided a new order-optimal method
  - A variant of SPIDER with random length of inner loop, making the algorithm easier to analyze

Probabilistic Gradient Estimator (PAGE)

The PAGE Algorithm

- Pick $\mathbf{x}_0$, step-size s, mini-batch sizes B and $B' < B$, probabilities $\{p_k\}_{k\geq 0} \in (0,1]$, number of iterations T
- Let $\mathbf{g}_0 = \frac{1}{B}\sum_{i\in I}\nabla f_i(\mathbf{x}_0)$, where I is a random mini-batch with $|I| = B$
- **for** $k = 0, 1, 2, \ldots, T-1$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s\mathbf{g}_k,$$

$$\mathbf{g}_{k+1} = \begin{cases} \frac{1}{B}\sum_{i\in I_k}\nabla f_i(\mathbf{x}_{k+1}), & \text{w.p. } p_k, \\ \mathbf{g}_k + \frac{1}{B'}\sum_{i\in I'_k}[\nabla f_i(\mathbf{x}_{k+1}) - \nabla f_i(\mathbf{x}_k)], & \text{w.p. } 1-p_k, \end{cases}$$

choose $s \leq \frac{1}{L(1+\sqrt{B/B'})}$, $B=N$.

where $|I_k| = B$ and $|I'_k| = B'$
end for

- **Output:** $\hat{\mathbf{x}}_T$ chosen uniformly from $\{\mathbf{x}_k\}_{k=1}^T$

$B' \leq \sqrt{B}$, $p_k = \frac{B'}{B'+B}$, then

sample complexity $O\left(\frac{2\Delta_0 L}{\varepsilon^2}\left(1+\sqrt{\frac{B}{B'}}\right)\right)$

$\leq N + \frac{8\Delta_0 L\sqrt{N}}{\varepsilon^2}$

$= O(N + \sqrt{N}\varepsilon^{-2})$.

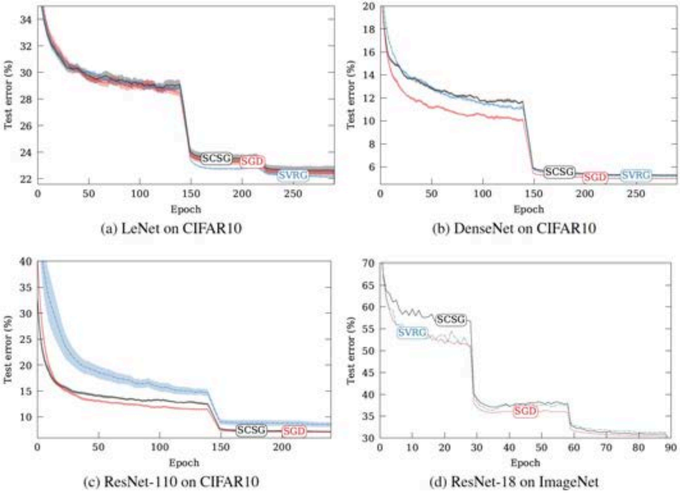
Summary of Sample Complexity Results for VR Methods

Method	References	Sample Complexity
Lower Bound	[Fang et al. NeurIPS'18]	$L\Delta_0 \min\{\sigma\epsilon^{-3}, \sqrt{N}\epsilon^{-2}\}$
GD		$NL\Delta_0\epsilon^{-2}$
SGD (bnd. var.)	[Ghadimi & Lan, SIAM-JO'13]	$L\Delta_0 \max\{\epsilon^{-2}, \sigma^2\epsilon^{-4}\}$
SGD (ubd. var.)	[Khaled & Richtarik, '20]	$\frac{L^2\Delta_0}{\epsilon^4} \max\{\Delta_0, \Delta_*\}$
SVRG ($B = 1$)	[Reddi et al. NeurIPS'16]	$NL\Delta_0\epsilon^{-2}$
SVRG ($B = \lceil N^{\frac{2}{3}} \rceil$)	[Reddi et al. NeurIPS'16]	$N^{\frac{2}{3}} L\Delta_0\epsilon^{-2}$
SAGA ($B = 1$)	[Reddi et al. NeurIPS'16]	$NL\Delta_0\epsilon^{-2}$
SAGA ($B = \lceil N^{\frac{2}{3}} \rceil$)	[Reddi et al. NeurIPS'16]	$N^{\frac{2}{3}} L\Delta_0\epsilon^{-2}$
SpiderBoost	[Wang et al. NeurIPS'19]	$N^{\frac{1}{2}} L\Delta_0\epsilon^{-2}$
SPIDER	[Fang et al. NeurIPS'18]	$L\Delta_0 \min\{\sigma\epsilon^{-3}, \sqrt{N}\epsilon^{-2}\}$
PAGE	[Li et al. ICML'21]	$L\Delta_0 \min\{\sigma\epsilon^{-3}, \sqrt{N}\epsilon^{-2}\}$

- Notation: $\Delta_0 = f(\mathbf{x}_0) - f^*$, $\Delta_* = \frac{1}{N}\sum_{i=1}^{N}(f^* - f_i^*)$, σ^2 is a uniform bound for the variance of stochastic gradient, B is batch size
- All results are for finite-sum with L-smooth summands. Sample complexity means the overall number of stochastic first-order oracle calls to find an ϵ-stationary point

Caveat of Variance-Reduced Methods

- In deep neural networks training, VR methods work typically worse than SGD or SGD+Momentum [Defazio & Bottou, NeurIPS'19]
  - Bad behavior of VR methods with several widely used deep learning tricks (e.g., batch normalization, data augmentation and dropout)



Next Class

First-Order Methods with Adaptive Learning Rates