Columbia University in the City of New York
Optimization Methods for Machine Learning
Instructors: Satyen Kale
Authors: Gurpreet Singh
Email: gurpreet.s@columbia.edu

SCRIBE
14

Reducing the Variance of Stochastic Gradients

1. Finite Sum Minimization

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \;=\; \frac{1}{n}\sum_{i=1}^{n} g_i(\mathbf{x})$$

where we assume the functions g_i are $\alpha-$strongly convex and β-smooth.

Vanilla gradient descent for this looks like the following

$$\mathbf{x}_{t+1} \;=\; \mathbf{x}_t - \eta\frac{1}{n}\sum_{i=1}^{n}\nabla g_i(\mathbf{x}_t)$$

With the above approach, we get ϵ-suboptimality in $\mathcal{O}\left(\frac{\beta}{\alpha}\log\left(\frac{\beta D^2}{2\epsilon}\right)\right)$ iterations. However, since we have n gradient evaluations in each iteration, each iteration is expensive.

We can view the same problem in the following way, however.

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) \;=\; \mathop{\mathbb{E}}_{i\sim[\,n\,]}\left[\,g_i(\mathbf{x})\,\right]$$

We use Stochastic Gradient Descent to get to the solution. In that case, we now need $\mathcal{O}\left(\frac{\beta(D^2+\sigma^2/\alpha^2)}{\epsilon}\right)$ iterations[1] to converge with ϵ-suboptimality. Even though the number of iterations increase by a huge factor, we only require *one* gradient computation in each iteration. As the sizes of datasets increase, the complexity of each iteration of full-batch gradient descent increases linearly. In such cases, stochastic gradient can perform much better.

However, SGD can often have very high variance σ which slows down converge. In order to find an optimal point between the low number of iterations in GD and the low cost of one iteration in SGD, we attempt to reduce the variance of each SGD step by using a lower variance estimator for the function $f(\mathbf{x})$. This is given as

$$f(\mathbf{x}) \;=\; \mathop{\mathbb{E}}_{\{i\}\sim[\,n\,]}\left[\sum_{r=1}^{k} g_{i_r}(\mathbf{x})\right]$$

[1] We are ignoring a $\log(1/\epsilon)$ factor here since it is a lower order term.

2. Lower Variance SGD using SAG

The problem with SGD is that we only use one function g_i out of the the n functions in each iteration which causes high variance gradient estimates. Stochastic Averaged Gradients (SAG) tries to lower the variance of each estimate without increasing the number of gradients used per iteration but only computing 2 gradients per iteration. This is achieved by book-keeping the previous gradient estimates and updating these estimates to be closer to the current data points. The algorithm for SAG is given in algorithm 1.

Algorithm 1: Stochastic Averaged Gradients

1. Maintain, for ever $i \in [\,n\,]$, the last computed gradient for g_i
2. Define $L(i) = \max_{s<t} \left\{ s \,\middle|\, i \text{ was sampled in iteration } s \right\}$
3. Sample $i_t \sim [\,n\,]$
4. Update
$$\mathbf{x}_{t+1} \;=\; \mathbf{x}_t - \frac{\eta}{n}\left[\nabla g_{i_t}(\mathbf{x}_t) - \nabla g_{i_t}\left(\mathbf{x}_{L(i_t)}\right) + \sum_{i=1}^{n} \nabla g_i(\mathbf{x}_{L(i)}) \right]$$
5. Set $L(i_t) = t$ and save the gradient $\nabla g_{i_t}(\mathbf{x}_t)$ for i_t.

Even though the variance of the gradient estimator is, in general, lower in SAG, the gradient estimator is biased. This can be easily analyzed by taking an expectation of the gradient estimator. That is, we have

$$\mathop{\mathbb{E}}_{i_t \sim [\,n\,]}\left[\frac{1}{n}\left(\nabla g_{i_t}(\mathbf{x}_t) - \nabla g_i(\mathbf{x}_{L(i_t)}) + \sum_{i=1}^{n} \nabla g_i(\mathbf{x}_{L(i)}) \right) \,\middle|\, i_0, i_1 \ldots i_{t-1} \right] \neq \nabla f(\mathbf{x})$$

Although this seems problematic for showing the convergence of the descent algorithm, the inventors of the SAG algorithm were able to prove the following the convergence bound:

$$\mathbb{E}\left[\, f(\mathbf{x}_T) - f(\mathbf{x}^*) \,\right] \;\leq\; \left(1 - \min\left\{ \frac{\alpha}{16\beta}, \frac{1}{8n} \right\}\right)^T \left[\frac{3}{2}(f_{\mathbf{x}_0} - f(\mathbf{x}^*)) + \frac{4\beta}{n} D^2 \right]$$

Therefore, in order to achieve ϵ-suboptimality, we would need $\mathcal{O}\left(\frac{\log\left(\frac{1}{\epsilon}\right)}{\min\left\{ \frac{\alpha}{16\beta}, \frac{1}{8n} \right\}} \right)$. Hence, we have

$$\begin{aligned} T \;&=\; \mathcal{O}\left(\max\left\{ \frac{16\beta}{\alpha}, 8n \right\} \log\left(\frac{1}{\epsilon}\right) \right) \\ &=\; \mathcal{O}\left(\left(\frac{16\beta}{\alpha} + 8n \right) \log\left(\frac{1}{\epsilon}\right) \right) \end{aligned}$$

This also equals (up to a factor of 2) the number of gradient evaluations. This represents a significant improvement over full-batch gradient descent or SGD, in cases when the condition numnber $\frac{\beta}{\alpha}$ is large.

3. SAGA

SAGA improves upon SAG by changing the gradient estimator in order to make it unbiased. The estimator used by SAGA is as follows

$$\nabla f(\mathbf{x}) \;=\; \mathop{\mathbb{E}}_{i_t \sim [\,n\,]} \left[\nabla g_{i_t}(\mathbf{x}_t) - \nabla g_i(\mathbf{x}_{L(i_t)}) + \frac{1}{n} \sum_{i=1}^{n} \nabla g_i(\mathbf{x}_{L(i)}) \,\middle|\, i_0, i_1 \dots i_{t-1} \right]$$

One can understand SAGA as fixing the bias of SAG gradients by subtracting the expected value of the gradients therefore making it unbiased. SAGA affords a slightly better convergence rate than SAG. Similar to SAG, for strongly convex and smooth subfunctions, SAGA attains ϵ-suboptimality in $\mathcal{O}\left(\left(\frac{3\beta}{\alpha} + 4n\right) \log\left(\frac{1}{\epsilon}\right)\right)$ iterations

4. Stochastic Variance Reduced Gradients

Stochastic Variance Reduced Gradients (SVRG) is yet another approach to gradient descent for finite sum minimizations. We study this because it's easier to analyse the convergence of SVRG over SAG and SAGA.

Algorithm 2: Stochastic Variance Reduced Gradients

1. Start with arbitrary $\mathbf{x}_0^{(0)}$
2. For $k = 0, 1 \dots K-1$
   (a) Set $\mathbf{x}_0 = \mathbf{x}_0^{(k)}$, compute $\nabla f(\mathbf{x}_0) = \frac{1}{n} \sum_{i=1}^{n} \nabla g_i(\mathbf{x}_0)$
   (b) For $t = 0, 1 \dots T-1$, update
   $$\mathbf{x}_{t+1} \;=\; \mathbf{x}_t - \eta \left(\nabla g_i(\mathbf{x}_t) - \nabla g_i(\mathbf{x}_0) + \nabla f(\mathbf{x}_0) \right)$$
   (c) Set $\mathbf{x}_0^{(k+1)} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}_t$
3. Output $x_0^{(K)}$.