10-725/36-725: Convex Optimization **Fall 2019**

Lecture 23: November 13

Lecturer: Ryan Tibshirani *Scribes: Ramkumar Natarajan and Tom Yan*

23.1 Stochastic gradient descent

Consider minimizing the average of a bunch of functions:

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

Recall that gradient descent goes as follows:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{n} \sum_{i=1}^n \nabla f_i(x^{(k-1)})$$

The stochastic gradient descent replaces the averaged gradient with the gradient of a randomly chosen function:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f_{i_k}(x^{(k-1)})$$

where i_k is randomly drawn from $\{1, ..., n\}$.

Note that:

$$\mathbb{E}[\nabla f_{i_k}(x^{(k-1)})] = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x^{(k-1)})$$

So the gradient used is an unbiased, albeit higher variance estimate of the gradient used for GD.

To reduce the variance, mini-batch stochastic gradient descent can be used, in which we sample $I_k \in [n]$ with $|I_k| = b$ functions and use the average of their gradients:

$$\mathbb{E}[\nabla f_{i_k}(x^{(k-1)})] = \frac{1}{b} \sum_{i \in I_k}^n \nabla f_i(x^{(k-1)})$$

Note that this is again an unbiased estimate of the gradient, b times more expensive than SGD but having $1/b$th the variance.

23.1.1 Example: Logistic Regression

Now let us compare the convergence rate and the running time of full gradient descent, stochastic gradient descent and mini-batch gradient descent for an example logistic regression problem. As mentioned before, the mini-batch stochastic gradient descent reduces the variance at the cost of expensive gradient update step. The update costs of these three methods are:

- Full gradient descent: $\mathcal{O}(np)$
- Stochastic gradient descent: $\mathcal{O}(p)$
- Mini-batch gradient descent: $\mathcal{O}(nb)$

where b is the size of the mini-batch and p is the length of the feature vector in the logistic regression problem.

The size of the example logistic regression is: Data size, $n = 10000$ and feature size $p = 20$. Following is a comparison of convergence rate, running times and suboptimality gap respectively among these three methods.

We make the following observations from the experimental results:

- In Fig. 23.1, looking at the convergence of mini-batch and stochastic gradient (mini-batch of batch size 1), regardless of the size of the batch, the criterion is going to bounce around the optimal value (or the value of convergence) for a while. This reflects the amount of variance in the gradient updates.
- In Fig. 23.2, we can see faster running times for smaller batch sizes. However, notice that once it reaches certain criterion value, the convergence trend is almost the same for every batch size. It is also noted that we do not converge to optimum for smaller batch sizes.
- This implies that in theory, we do not obtain a lot of value by using larger batches if we are using diminishing step sizes. However, in many settings the batch-update can be parallelized to save compuation.

23.1.2 Convergence rates

Condition	GD Rate	SGD Rate
Convex	$O(1/\sqrt{k})$	$O(1/\sqrt{k})$
+ Lipschitz Gradient	$O(1/k)$	$O(1/\sqrt{k})$
+ Strongly Convex	$O(\gamma^k)$	$O(1/k)$

From the table, we can observe that there is no improvement in convergence for SGD even if the function has Lipschitz gradient and only provides sublinear convergence even under the strong convex assumption as opposed to linear convergence for Gradient descent. The following techniques below use variance reduction to modify SGD to take advantage of additional conditional constraints like Lipschitz gradient so as to match the convergence rates of GD.

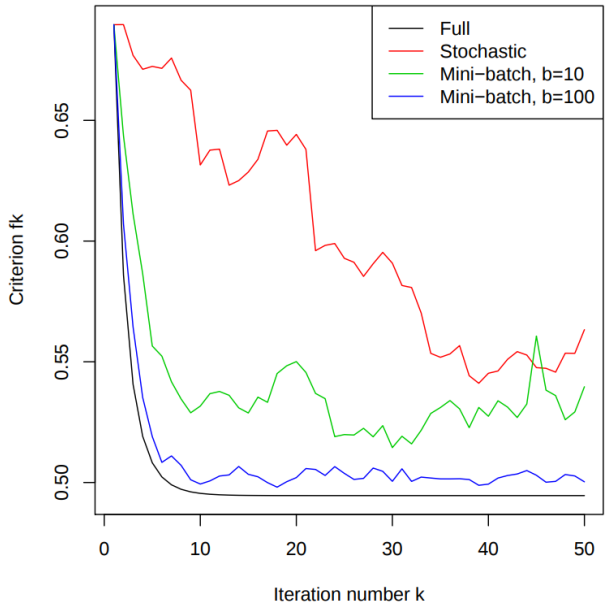

Figure 23.1: Rate of convergence: Criterion value vs Number of iterations

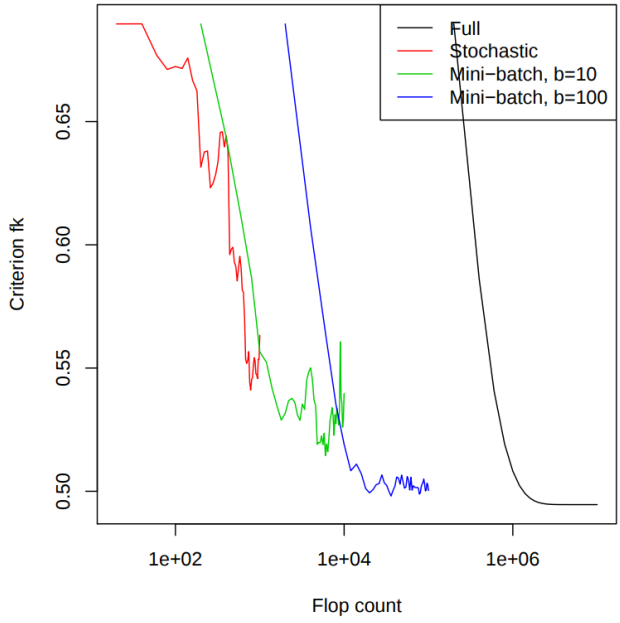

Figure 23.2: Running time: Criterion value vs Number of flops

23.2 Variance reduction (SAG, SAGA)

Until the year 2014 it was believed that the SGD cannot improve beyond the sublinear convergence even with Lipschitz gradient and strongly convex objective function as Nemirovski and others had established a

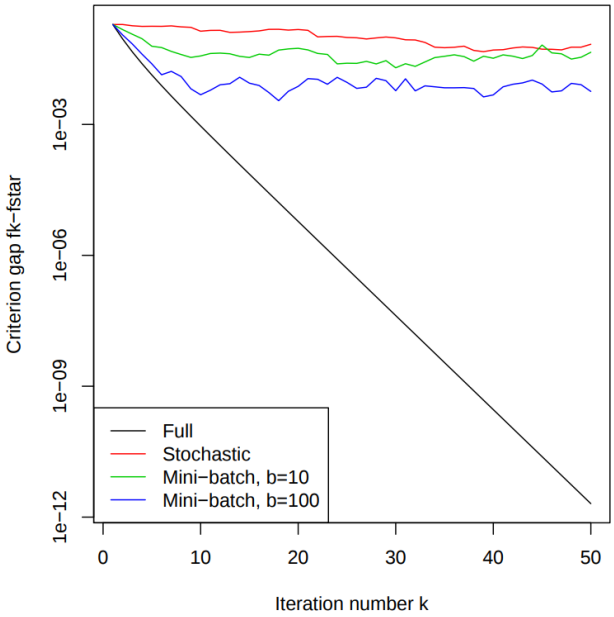

Figure 23.3: Suboptimality: Optimality gap vs Number of iterations

matching lower bound. However, this lower bound was for a general stochastic problem of the form:

$$f(x) = \int F(x, \xi) dP(\xi)$$

which describes the expectation with respect to χ over a measure P. But many problems encountered in machine learning have finite sums formulation ($f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$) based on which we described the stochastic gradient descent in Sec. 1. This observation lead to a series of modifications to SGD that can improve the convergence for Lipschitz gradient and strongly convex objective functions only with **finite sums representation**. We will describe two important algorithms of this class: Stochastic Average Gradient (SAG) and SAGA.

23.2.1 Stochastic average gradient

- Maintain a table, containing gradient g_i of f_i, for $i = 1, ..., n$
- Initialize $x^{(0)}$ and $g_i^{(0)} = \nabla f_i(x^{(0)})$, for $i = 1, ..., n$
- At steps $k = 1, 2, 3...$ pick random $i_k \in \{1, ..., n\}$ and set:

  $$g_{i_k}^{(k)} = \nabla f_{i_k}(x^{(k-1)})$$

  while setting $g_i^{(k)} = g_i^{(k-1)}$ for $i \neq i_k$.
- Update

  $$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{n} \sum_{i=1}^{n} g_i^{(k)}$$

SAG gradient estimates are no longer unbiased, but they have greatly reduced variance.

For the update step, at each iteration, it only requires an $O(1)$ addition for the aggregate gradient from the previous step to be modified to become the aggregate gradient for the current step:

$$x^{(k)} = x^{(k-1)} - t_k \cdot (\frac{g_{i_k}^{(k)}}{n} - \frac{g_{i_k}^{(k-1)}}{n} + \frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)})$$

The above update can be seen as a moving average over a sliding window which is a constant time calculation.

Suppose for each f_i, ∇f_i is Lipschitz with constant L. Write $\bar{x}^{(k)} = \frac{1}{k}\sum_{l=0}^{k-1} x^{(l)}$, then we have the below theorem for convergence rate:

Theorem 23.1 ***cite****: SAG, with a fixed step size* $t = \frac{1}{16L}$*, and initialization* $g_i^{(0)} = \nabla f_i(x^{(0)}) - \nabla f(x^{(0)})$ *for* $i = 1, ..., n$ *is such that:*

$$\mathbb{E}[f(\bar{x}^{(k)})] - f^\star \leq \frac{48n}{k}(f(x^{(0)}) - f^\star) + \frac{128L}{k}||x^{(0)} - x^\star||_2^2$$

This gives $O(1/k)$ convergence rate which matches that for GD. The only difference is that the constant factor for GD is $O(L)$ where it is $O(n + L)$ for SAG. Notice that the initialization $g_i^{(0)}$ centers the gradients around zero by removing the average from every element of the gradient; in other words, the average of $g_i^{(0)}$ is 0. Moreover, the first term in the above convergence bound depends on n; the authors of the work have suggested mitigating this by warm starting with the result of n SGD steps.

Furthermore, under the even stronger assumption of strong convexity where each f_i is strongly convex with parameter m:

Theorem 23.2 ***cite****: SAG, with a fixed step size* $t = \frac{1}{16L}$*, and initialization* $g_i^{(0)} = \nabla f_i(x^{(0)}) - \nabla f(x^{(0)})$ *for* $i = 1, ..., n$ *is such that:*

$$\mathbb{E}[\bar{x}^{(k)}] - f^\star \leq (1 - \min\{\frac{m}{16L}, \frac{1}{8n}\})^k \cdot (\frac{3}{2}f(x^{(0)}) - f^\star) + \frac{4L}{n}||x^{(0)} - x^\star||_2^2)$$

This gives $O(\gamma^k)$ convergence rate and again matches that for GD. The convergence analysis proofs for SAG is extremely complicated because of the biased estimator in SAG update rule.

23.2.2 SAGA

- Maintain a table, containing gradient g_i of f_i, for $i = 1, ..., n$
- Initialize $x^{(0)}$ and $g_i^{(0)} = \nabla f_i(x^{(0)})$, for $i = 1, ..., n$
- At steps $k = 1, 2, 3...$ pick random $i_k \in \{1, ..., n\}$ and set:

  $$g_{i_k}^{(k)} = \nabla f_{i_k}(x^{(k-1)})$$

  while setting $g_i^{(k)} = g_i^{(k-1)}$ for $i \neq i_k$.

- Update

$$x^{(k)} = x^{(k-1)} - t_k \cdot (g_{i_k}^{(k)} - g_{i_k}^{(k-1)} + \frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)})$$

Notice that the only difference between SAG and SAGA is the heavier weight on the updated gradient at step k. We have:

$$[g_{i_k}^{(k)} - g_{i_k}^{(k-1)}] + \frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)}$$

instead of

$$[\frac{g_{i_k}^{(k)}}{n} - \frac{g_{i_k}^{(k-1)}}{n}] + \frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)}$$

Interestingly, the SAGA gradient is unbiased. This is because

$$\mathbb{E}[g_{i_k}^{(k-1)}] = \frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)}$$

and

$$\mathbb{E}[g_{i_k}^{(k)}] = \mathbb{E}[\nabla f_{i_k}(x^{(k-1)})] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x^{(k-1)})$$

SAGA matches convergence rates of SAG. In fact, a following simple calculation can show that the SAGA style update gives an unbiased estimate for such algorithms; consider a family of estimators for $\mathbb{E}(X)$

$$\theta_\alpha = \alpha(X - Y) + \mathbb{E}(Y)$$

where $\alpha \in [0, 1]$ and X, Y are assumed to be correlated. The first two moments, mean and variance, can be calculated as follows for θ_α

$$\begin{aligned}\mathbb{E}(\theta_\alpha) &= \alpha\mathbb{E}(X) - \alpha\mathbb{E}(Y) + \mathbb{E}(Y) \\ &= \alpha\mathbb{E}(X) + (1-\alpha)\mathbb{E}(Y)\end{aligned}$$

$$\begin{aligned}Var(\theta_\alpha) &= \alpha^2 Var(X - Y) \\ &= \alpha^2(Var(X) + Var(Y) - 2Cov(X, Y))\end{aligned}$$

SAGA uses $\alpha = 1$ (unbiased) and SAG uses $\alpha = 1/n$ (biased). Thus, SAGA has a higher variance than SAG but is unbiased.

23.2.3 Empirical Comparison of techniques

Again, let us take the same logistic regression example and compare SGD with the newly introduced SAG and SAGA. In order to remove bias due to initialization, the experiment is run over 30 initialization and averaged.

The most astonishing observation is perhaps the shockingly low variance of SAG which almost looks deterministic. It can be observed that SAGA converges better than SAG. Although the results demonstrate significant improvement over SGD, these methods required the below setup to obtain this performance.

- Warm starting with SGD helped a lot (one pass over the data with one full cycle of SGD). From this, we obtain an initialization $\beta^{(0)}$ to start SGD, SAG and SAGA.
- Recall that we centered the gradients around 0 for SAG. However, for SAGA centering the gradients exacerbated the convergence and better results were in fact obtained by simply initializing them at zeros. For the results SAGA was initialized at $g_i^{(0)} = \nabla f_i(\beta^{(0)})$
- Tuning the fixed step sizes for SAGA was fine; seemingly on par with tuning for SGD, and more robust than tuning for SAG
- Interestingly, the SAGA criterion curves look like SGD curves except that they are jagged and highly variable. While SAG looks very different, and this really emphasizes the fact that its update have much lower variance.

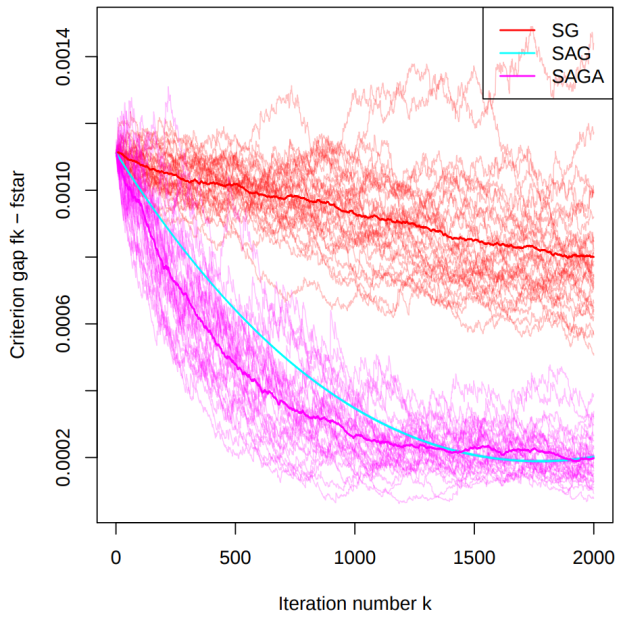

Figure 23.4: Comparison between SGD, SAG and SAGA on a logistic regression example.

23.3 Other Stochastic Methods

Since the year 2014, several other methods with ideas derived from SAG and SAGA have been developed for finite sums objective function representation. Some important ones include:

- SDCA (Shalev-Schwartz, Zhang, 2013): This method applies coordinate ascent to the dual of ridge regularized problems, and uses randomly selected coordinates. Effective primal updates are similar to SAG/SAGA.
- SVRG (Johnson, Zhang, 2013): This method is similar to SAG and SAGA, but does not store a full table of gradients. It just stores the average and updates them occasionally.
- Other methods include S2GD (Konecny, Richtarik, 2014), MISO (Mairal, 2013) and Finito (Defazio, Caetano, Domke, 2014), etc.
- A very nice review that discuss the connection between these methods are provided by SAG and SAGA papers.

23.4 Acceleration and momentum

For a strong convex setting, both SAG and SAGA have an iteration complexity of

$$\mathcal{O}((n + \frac{L}{m})log(\frac{1}{\epsilon}))$$

with a lower bound of

$$\mathcal{O}((n + \sqrt{\frac{nL}{m}})log(\frac{1}{\epsilon}))$$

However, we can do better than this by using acceleration ((Lan and Zhou 2015, Lin et al., 2015). Finite sums representation can be solved by this combination of Variance reduction and acceleration. Nevertheless, in convex problems, if we move the criterion from finite sum to general stochastic settings, $f(x) = \mathbb{E}_\xi(F(x.\xi))$, the performance of SGD does no improve.

For non-convex problems, reducing the variance is thought of as a wrong idea as variance is often necessary to escape from local minima and saddle points. But the idea of momentum instead of acceleration is a popular alternative. A classic approach (almost two decades older than Nesterov's acceleration) is Polyak's heavy ball method. This method works very well in practise but can also be fragile sometimes. The momemtum update rule is:

$$x^{(k)} = x^{(k-1)} + \alpha(x^{(k-1)} - x^{(k-2)}) - t_k \nabla f_{i_k}(x^{(k-1)})$$

The only difference is the term $\alpha(x^{(k-1)} - x^{(k-2)})$ that pushes the gradient in the direction of consequtive previous difference of steps. An example of Polyak's heavy ball vs. Nesterov's acceleration in optimizing a convex quadratic is provided in Shi et al., 2018.

23.5 Adaptive step sizes (AdaGrad)

From the discussion earlier, the step size have to be diminished over time for SGD to ensure convergence (we descried the bouncing convergence close to optimality in the convergence plots). A problem with this is that features which do not provide a rich gradient signal will soon not receive any more update due to diminishing

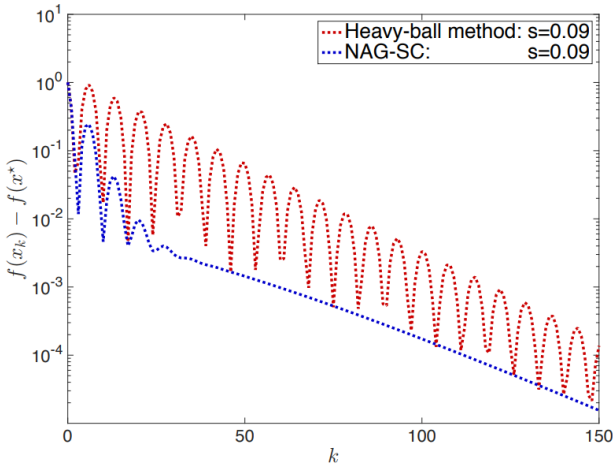

Figure 23.5: Comparison of Polyak's heavy ball (momentum) and Nesterov's acceleration.

step sizes. A possible solution is to adaptively change the step size based on how large the gradients were for the respective feature dimension for prior time-steps.

One popular adaptive step size method is called *AdaGrad*. Let $g^{(k)} = \nabla f_{i_k}(x^{k-1}))$, for $j = 1, ..., p$ do:

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{\sum_{l=1}^{k} (g_j^{(l)})^2}}$$

The advantage of the above update rule is that, we do not need to tune our learning rate anymore as α is now a fixed hyperparameter. It is noted that in sparse problems, AdaGrad performs much better than standard SGD. Several extensions of AdaGrad exists, *viz.,* Adam, RMSProp etc.

References

1. Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of Machine Learning Research 12.Jul (2011): 2121-2159.

2. Defazio, Aaron, Francis Bach, and Simon Lacoste-Julien. "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives." Advances in neural information processing systems. 2014.

3. Lan, Guanghui, and Yi Zhou. "An optimal randomized incremental gradient method." Mathematical programming 171.1-2 (2018): 167-215.

4. Nemirovski, Arkadi, et al. "Robust stochastic approximation approach to stochastic programming." SIAM Journal on optimization 19.4 (2009): 1574-1609.

5. Schmidt, Mark, Nicolas Le Roux, and Francis Bach. "Minimizing finite sums with the stochastic average gradient." Mathematical Programming 162.1-2 (2017): 83-112.